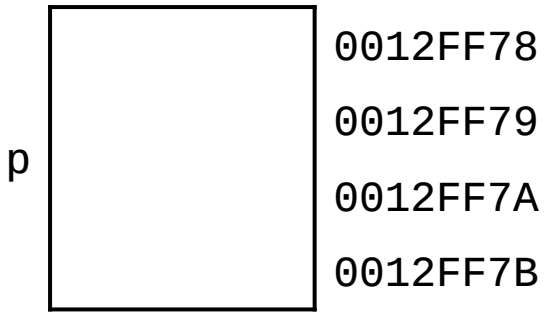
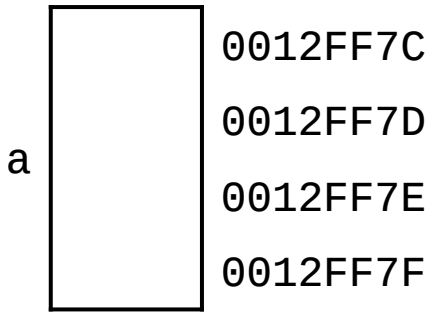
Pointers

Pointer Variable Declarations and Initialization

- Pointer variables contain memory addresses as their values

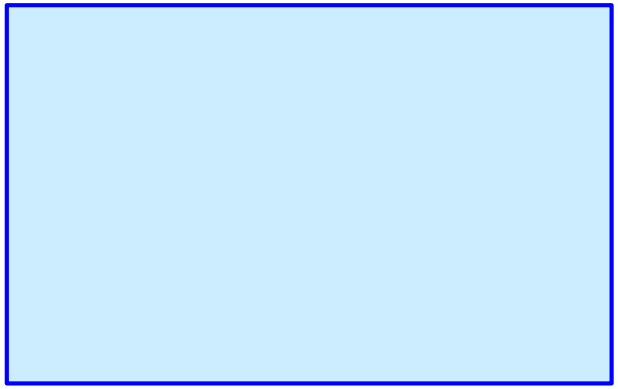
```cpp
#include <iostream>
using namespace std;

int main()
{
   int a;
   int *p; // p is a pointer to an integer

   a = 7;
   p = &a; // assign the address of a to p

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

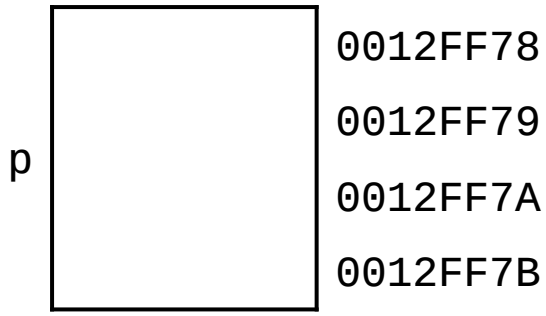


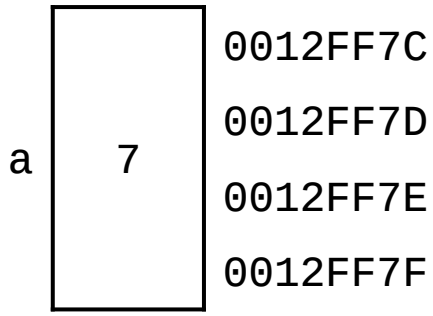



```
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

Output

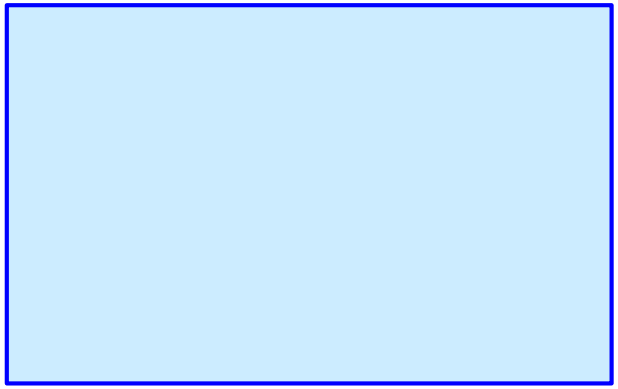


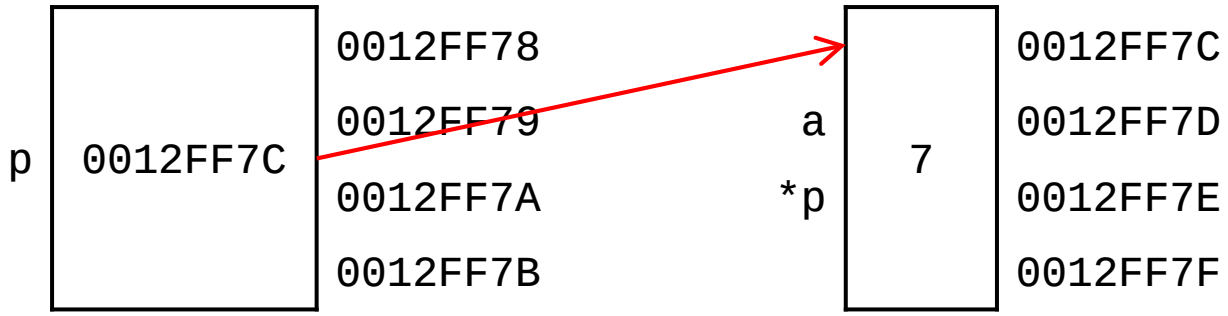





```
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

Output

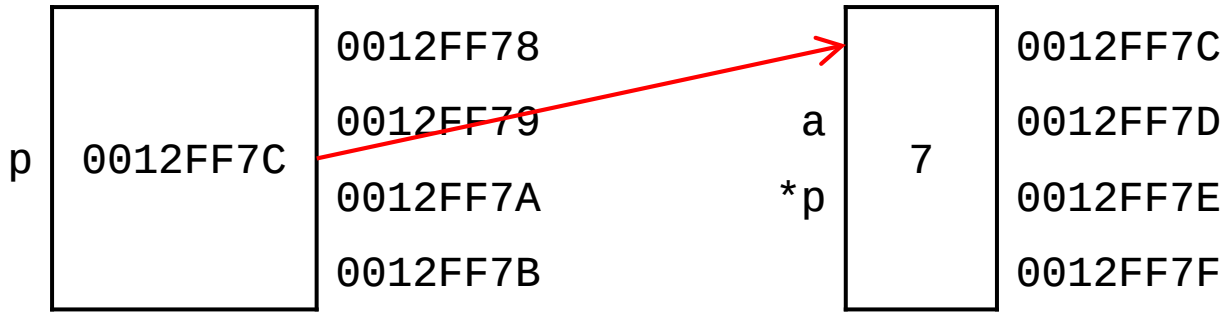




p points to a

```
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

Output

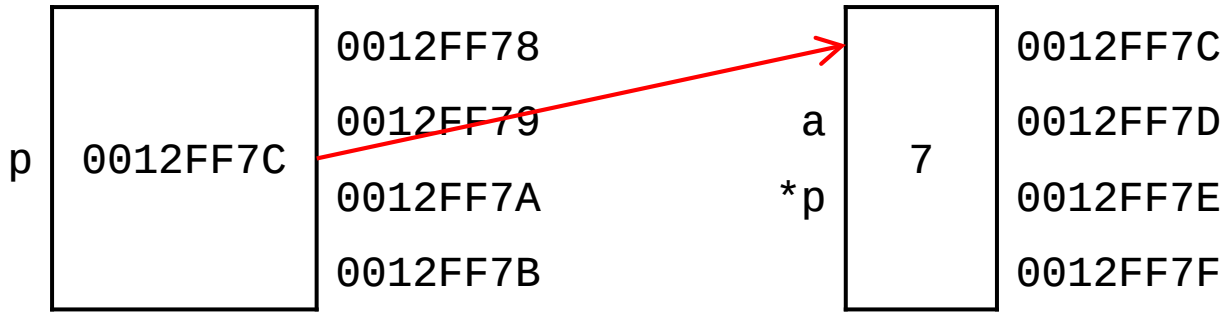


p points to a

```
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

Output

```
0012FF7C 0012FF7C
```

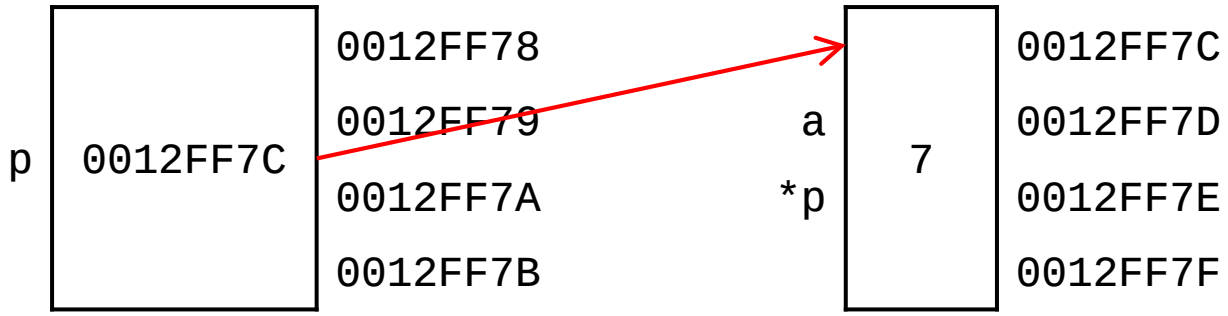


p points to a

```
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

Output

```
0012FF7C 0012FF7C
7 7
```

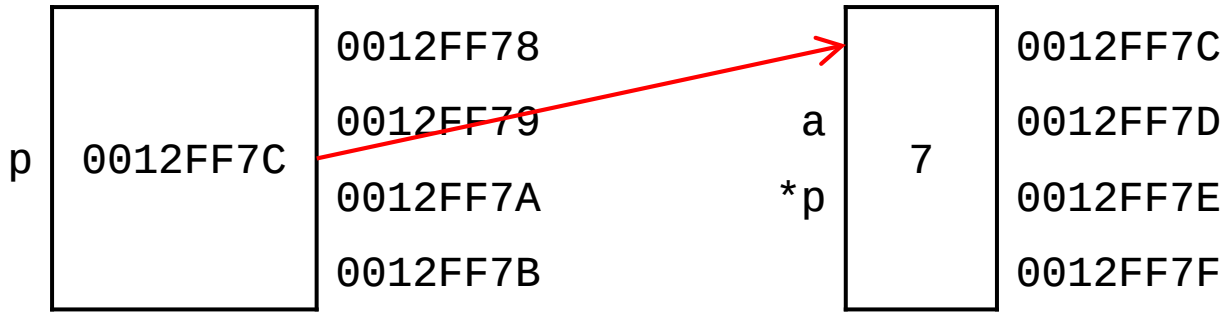


p points to a

```
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

Output

```
0012FF7C 0012FF7C
7 7
0012FF78
```



p points to a

```
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

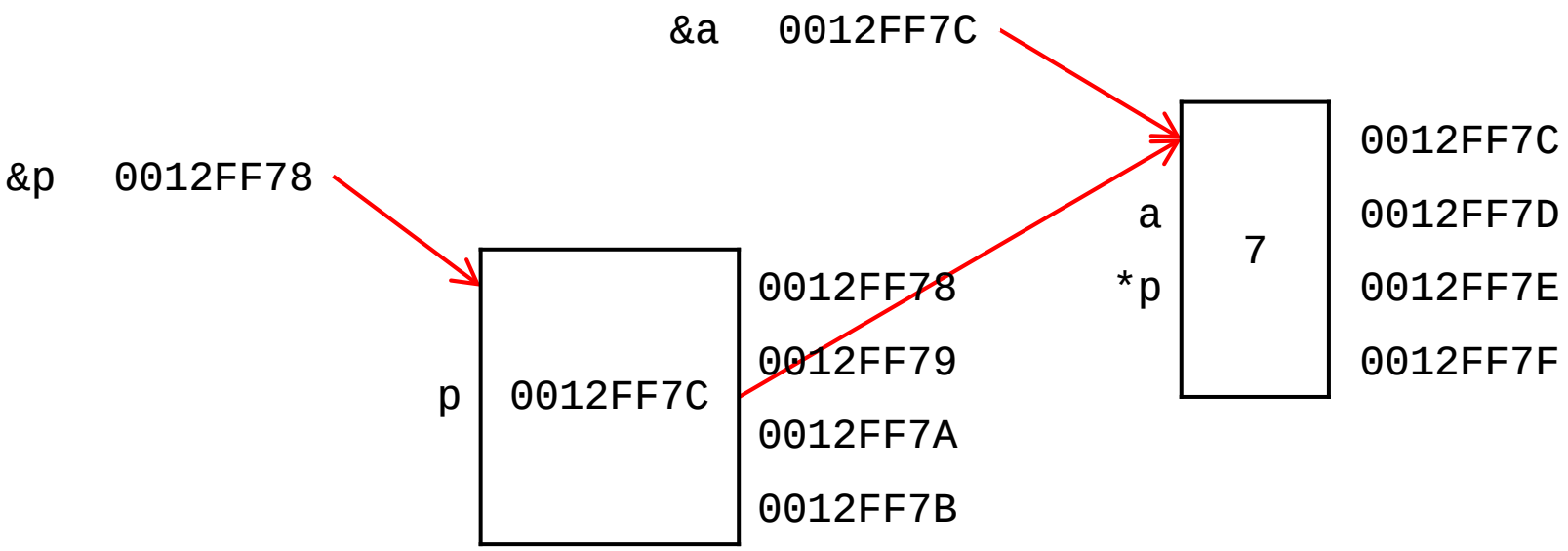
Output

```
0012FF7C 0012FF7C
7 7
0012FF78
0012FF7C 0012FF7C
```

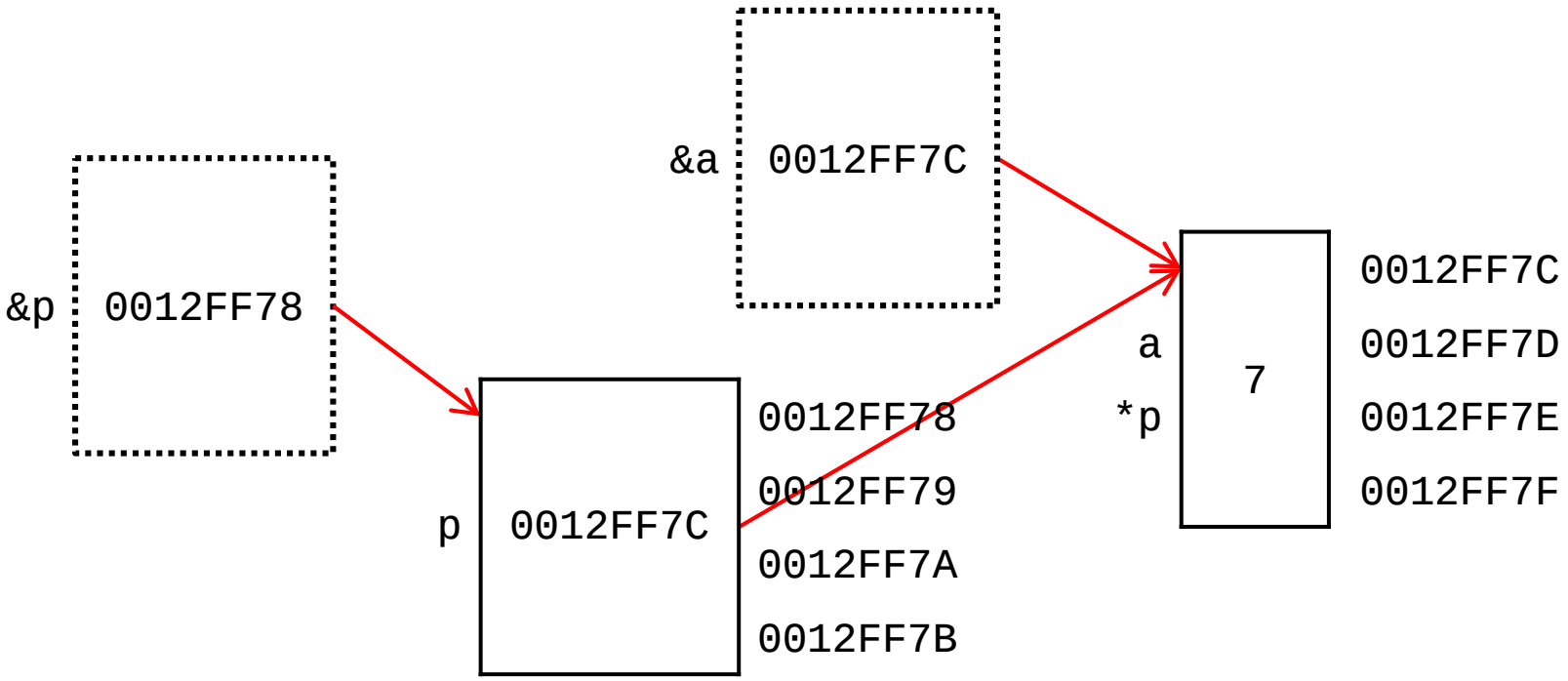
```cpp
int main()
{
   int a;
   int *p;

   a = 7;
   p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

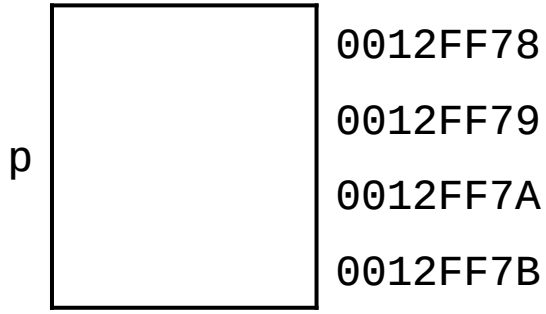
```cpp
int main()
{
   int a = 7;
   int *p = &a;

   cout << &a << " " << p << endl;
   cout << a << " " << *p << endl;
   cout << &p << endl;
   cout << &*p << " " << *&p << endl;
}
```

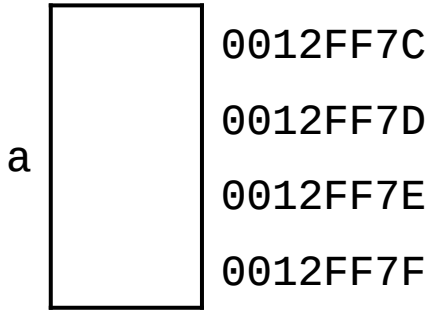
```
int a = 7;
int *p = &a;
cout << &*p << " " << *&p << endl;
```



```
int a = 7;
int *p = &a;
cout << &*p << " " << *&p << endl;
```

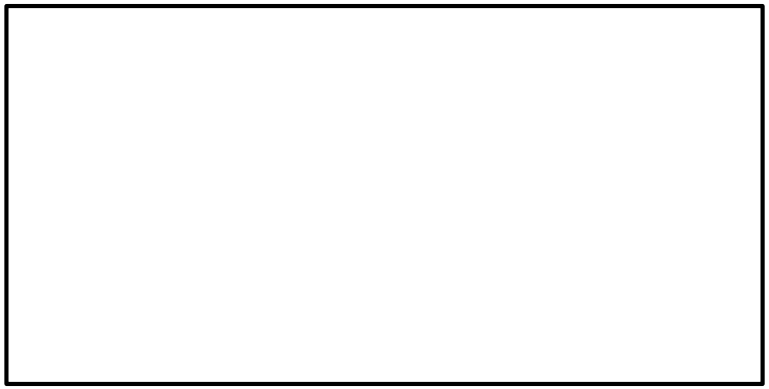


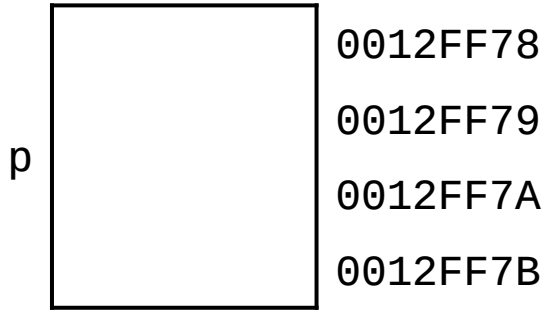


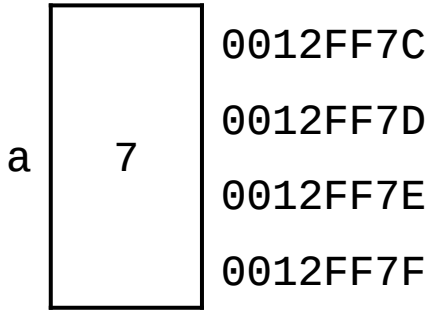



```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

Output

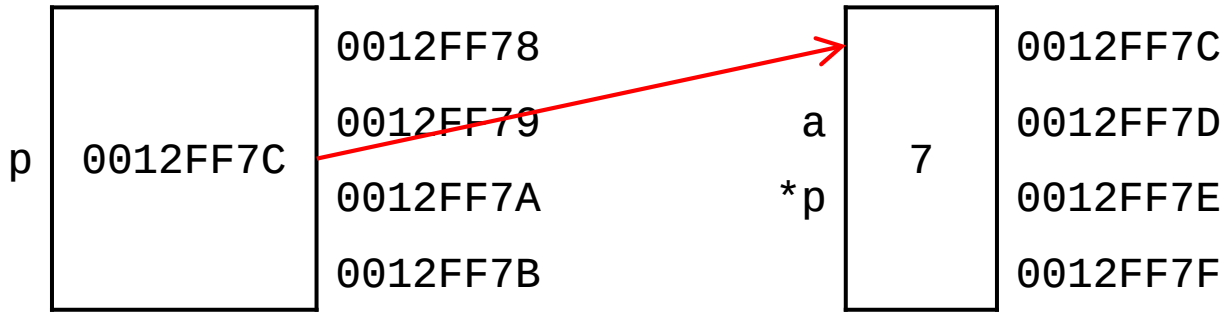







```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

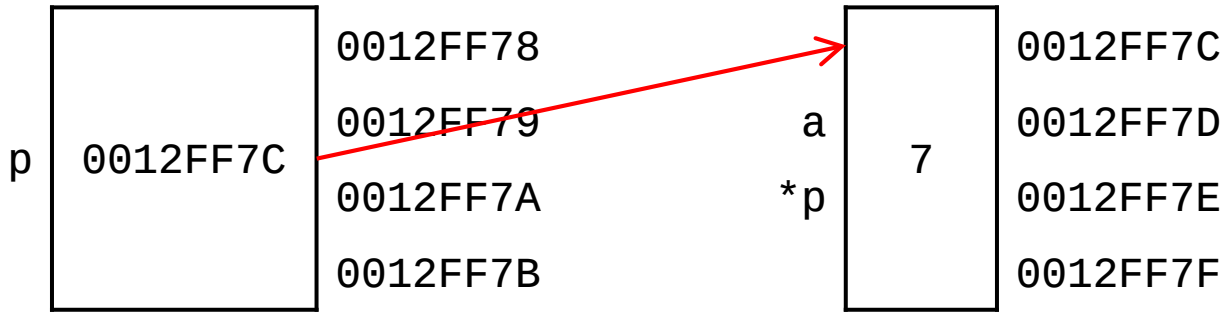
Output



p points to a

```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

Output

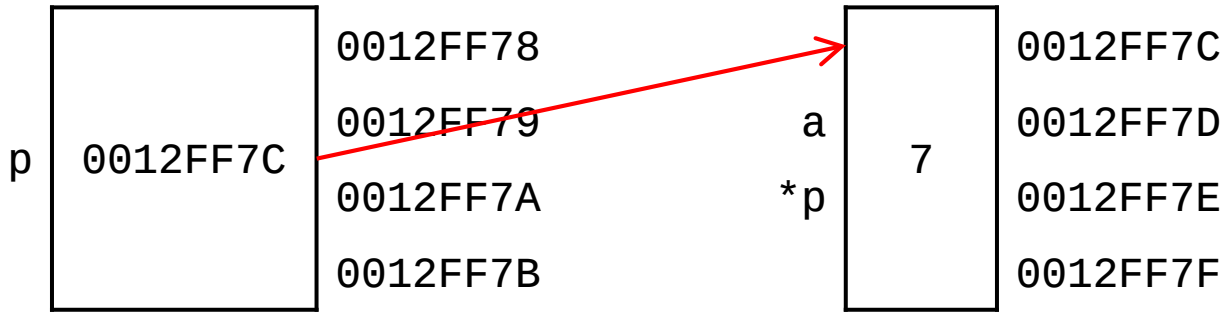


p points to a

```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

Output

```
7
```

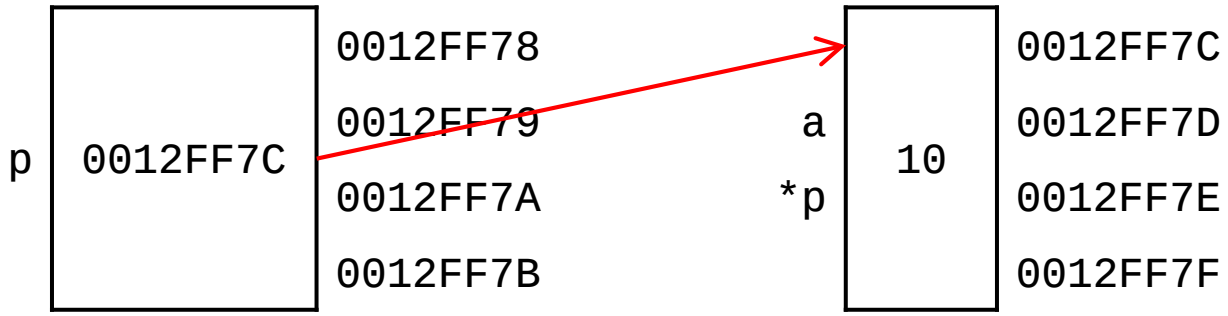


p points to a

```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

Output

```
7 7
```

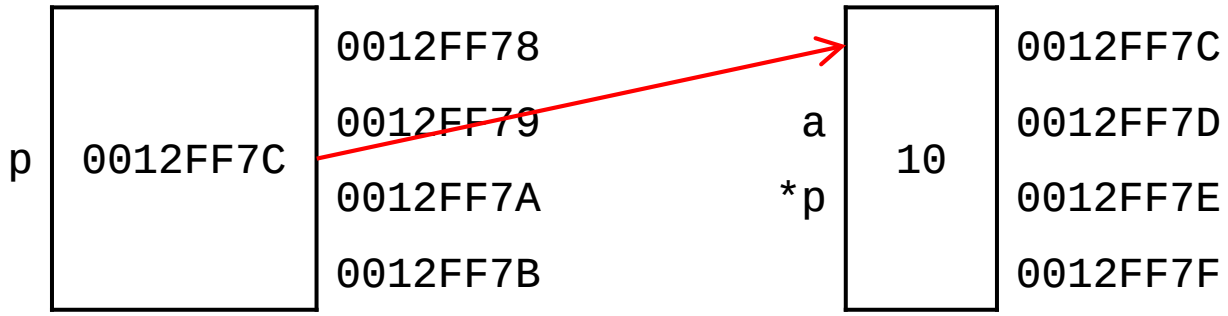


p points to a

```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

Output

```
7 7
```

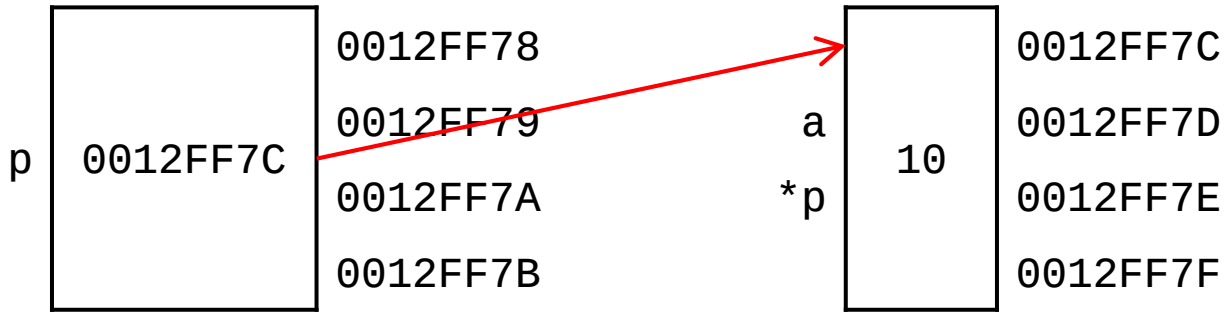


p points to a

```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

Output

```
7 7
10
```



p points to a

```
int a = 7;
int *p = &a;
cout << a << " ";
cout << *p << "\n";
*p = 10;
cout << a << " ";
cout << *p << "\n";
```

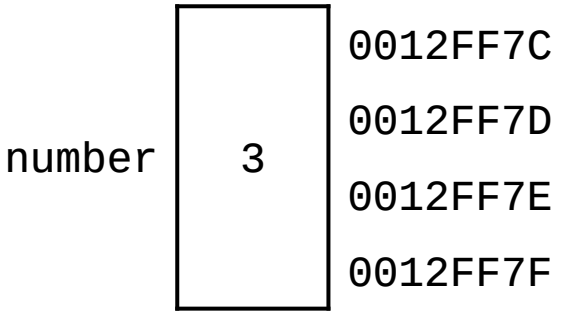
Output

```
7 7
10 10
```

Three ways to pass arguments to function

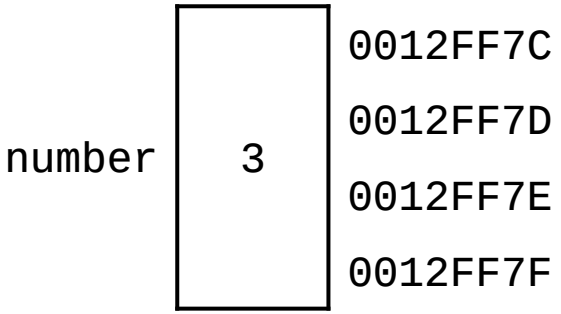
- Pass-by-value
- Pass-by-reference
- Pass-by-address

```
#include <iostream>
using namespace std;

void cube( int *n );

int main()
{
   int number = 3;

   cout << number << endl;

   cube( &number );

   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

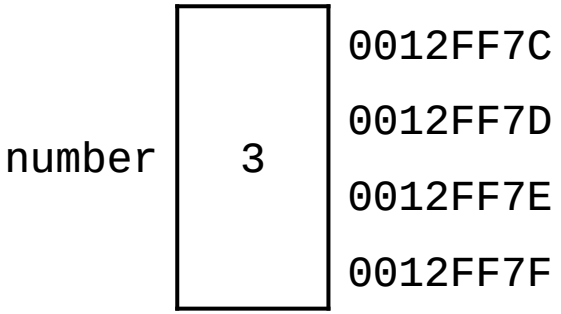
Output




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

Output

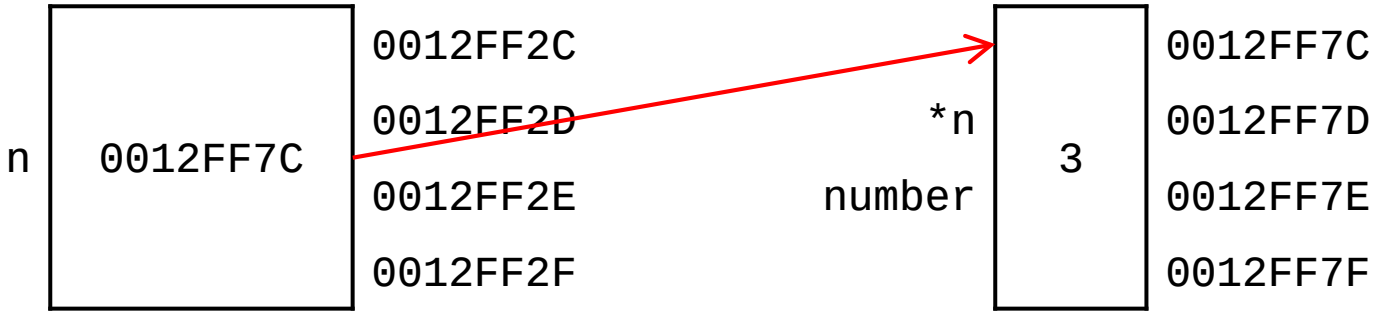
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

Output

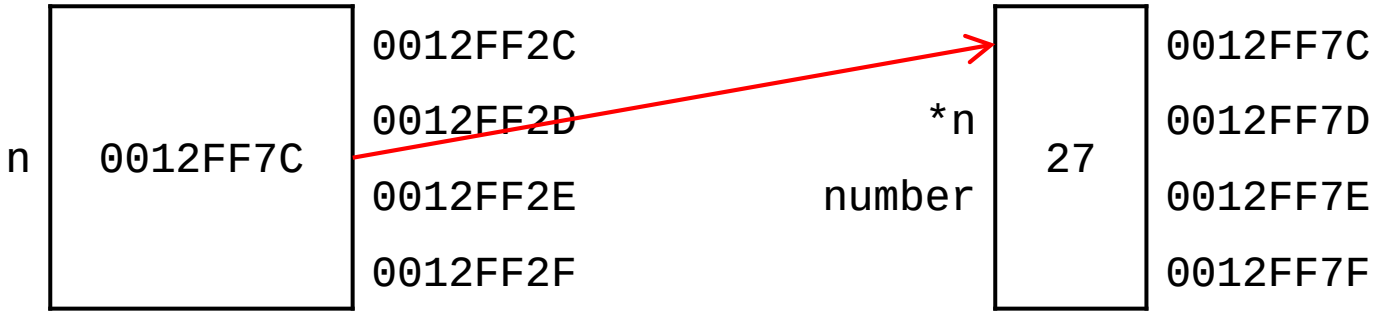
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

Output

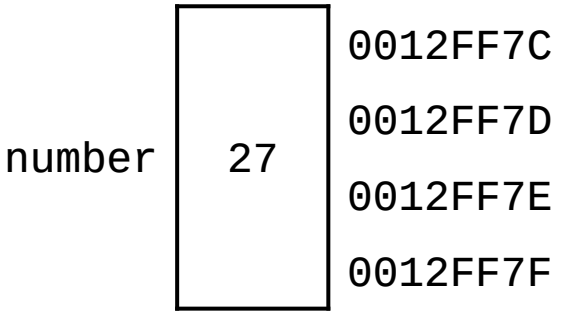
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

Output

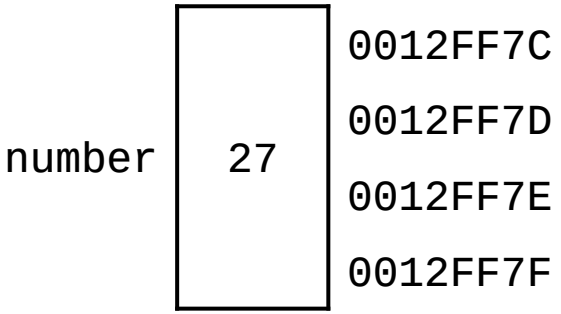
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

Output

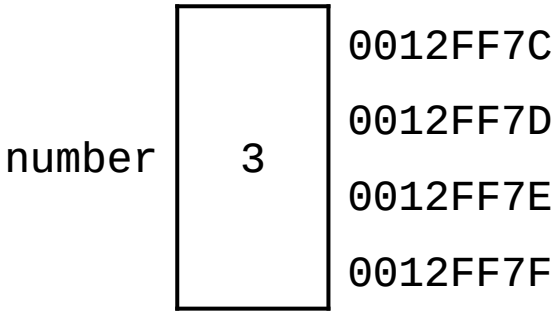
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

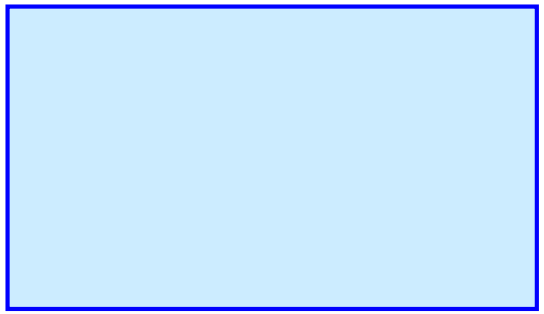
Output

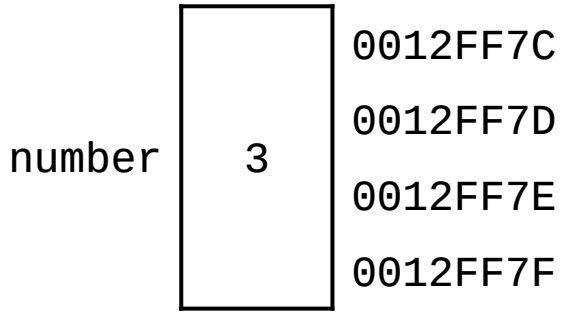
```
3
27
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   int *n = &number;
   *n = *n * *n * *n;
   cout << number << endl;
}
```

Output

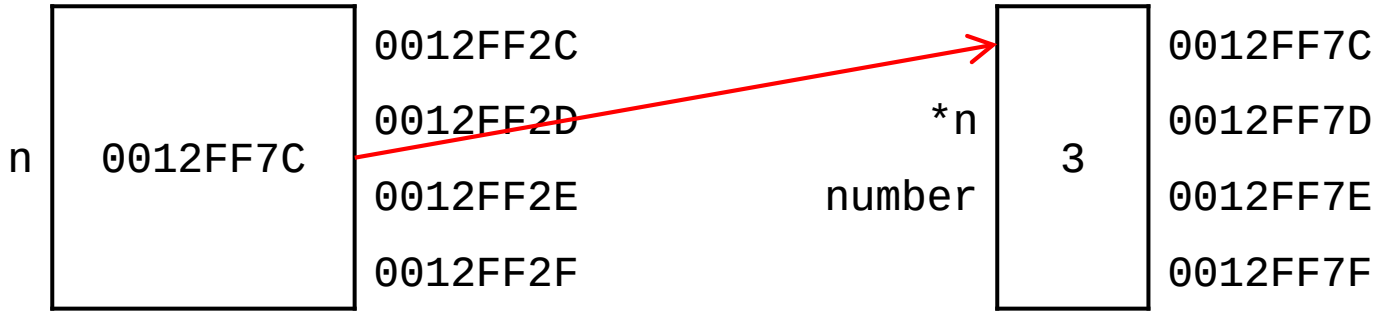





```
int main()
{
   int number = 3;
   cout << number << endl;
   int *n = &number;
   *n = *n * *n * *n;
   cout << number << endl;
}
```

Output

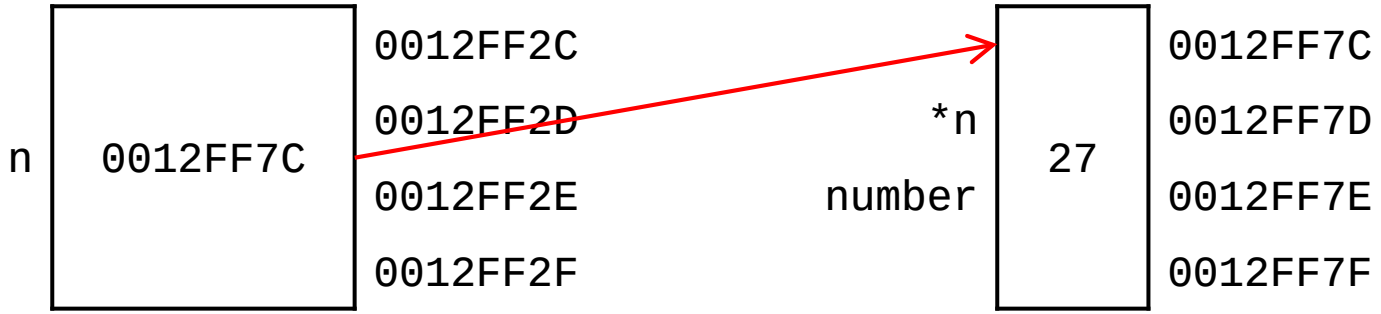
```
3
```




```
int main()
{
    int number = 3;
    cout << number << endl;
    int *n = &number;
    *n = *n * *n * *n;
    cout << number << endl;
}
```

Output

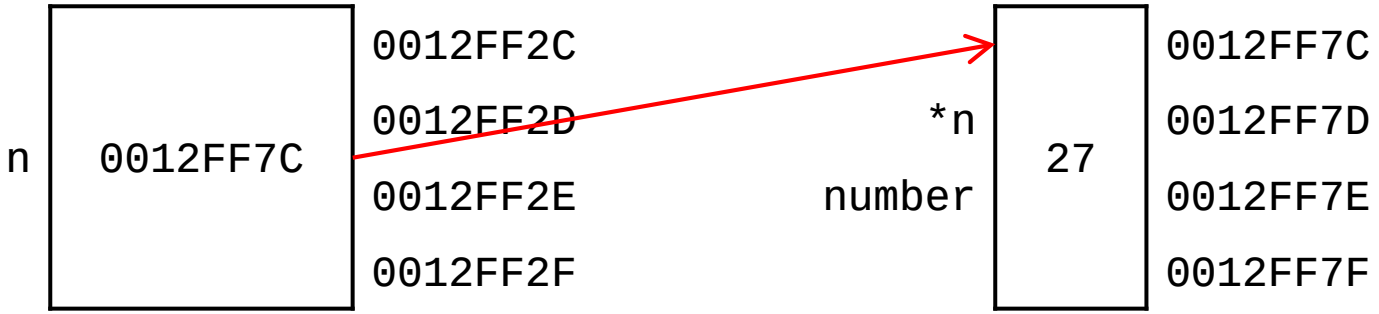
```
3
```




```
int main()
{
    int number = 3;
    cout << number << endl;
    int *n = &number;
    *n = *n * *n * *n;
    cout << number << endl;
}
```

Output

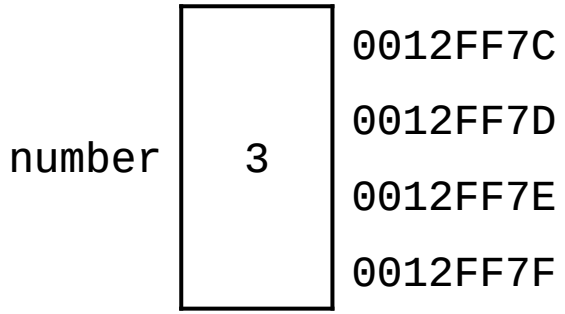
```
3
```




```
int main()
{
    int number = 3;
    cout << number << endl;
    int *n = &number;
    *n = *n * *n * *n;
    cout << number << endl;
}
```

Output

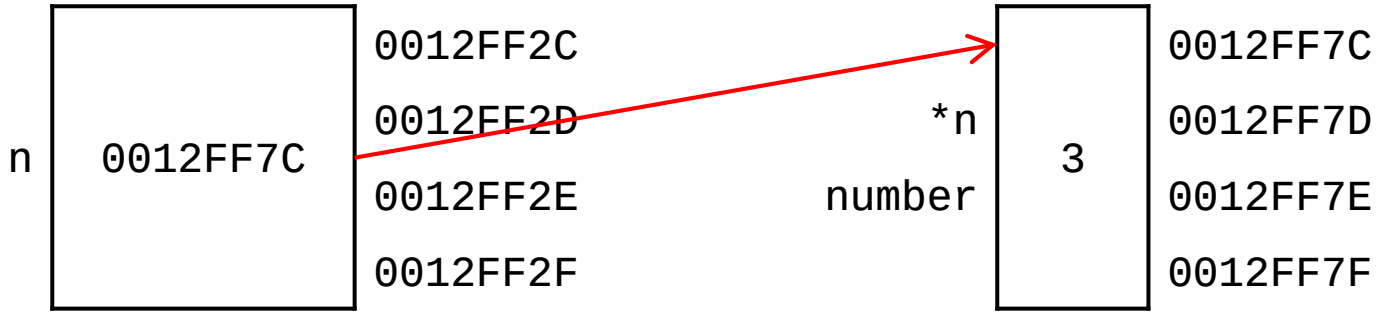
```
3
27
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   int *n = &number;
   *n = *n * *n * *n;
   cout << number << endl;
}
```

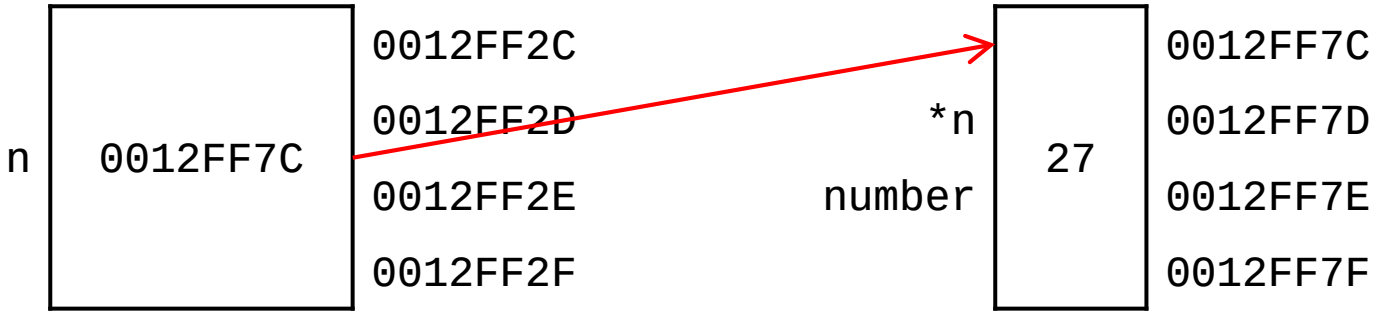
Comparison




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   int *n = &number;
   *n = *n * *n * *n;
   cout << number << endl;
}
```

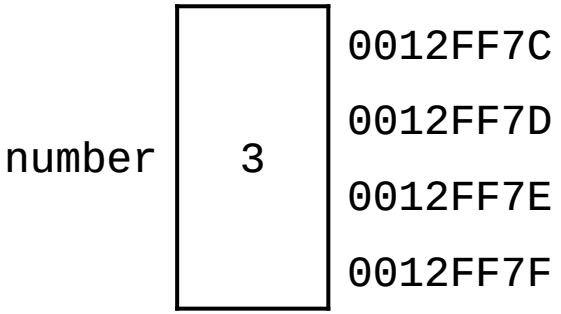
Comparison




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

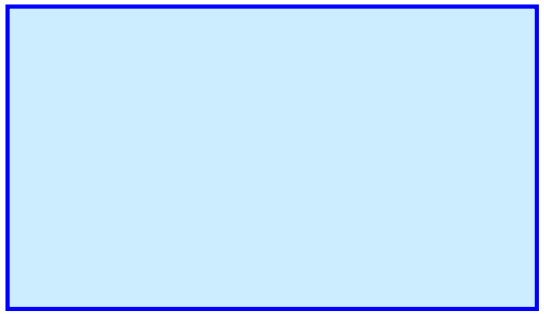
```
int main()
{
   int number = 3;
   cout << number << endl;
   int *n = &number;
   *n = *n * *n * *n;
   cout << number << endl;
}
```

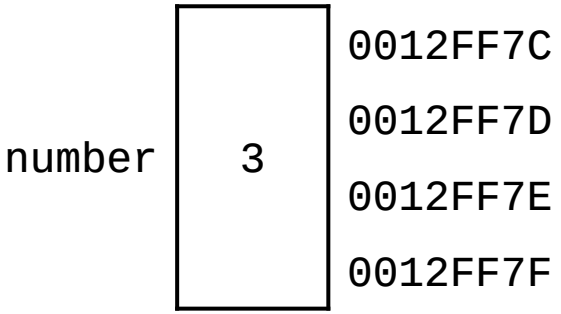
Comparison




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Output

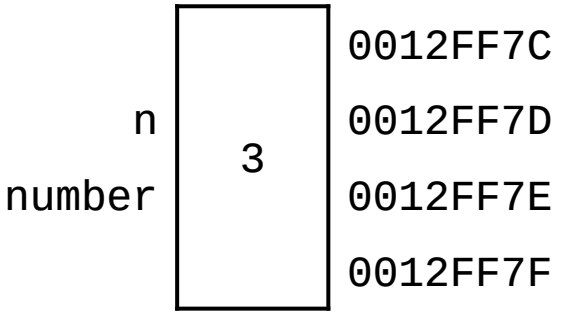





```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Output

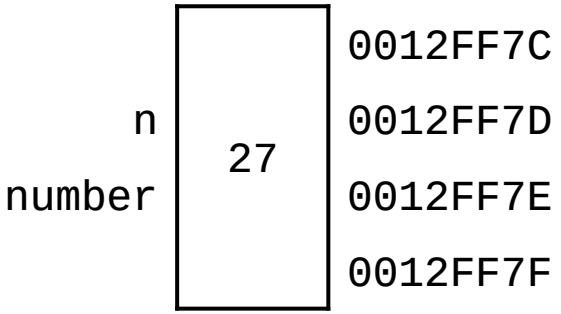
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Output

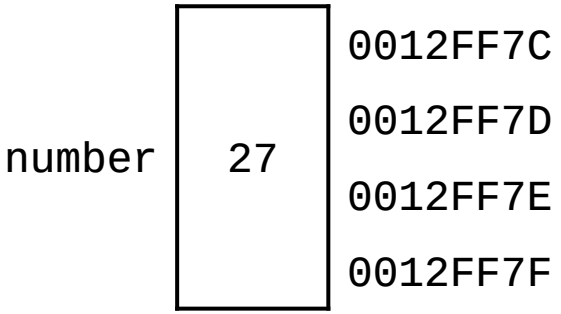
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Output

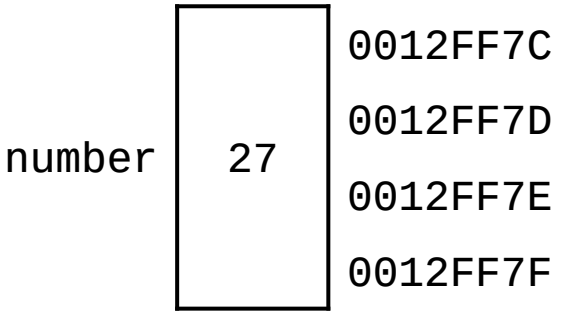
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Output

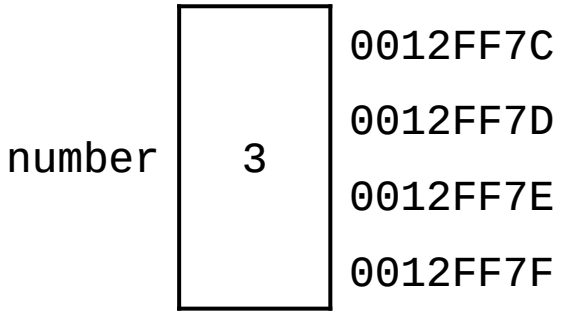
```
3
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

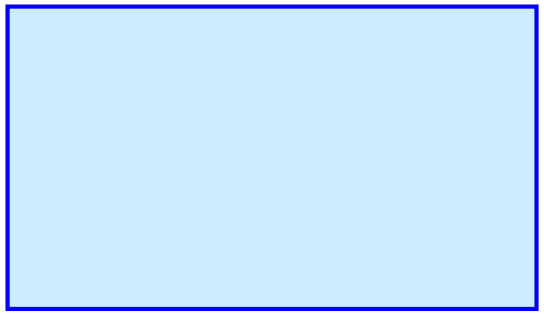
Output

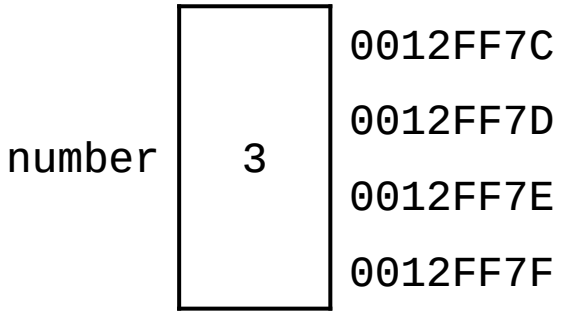
```
3
27
```




```
int main()
{
   int number = 3;
   cout << number << endl;
   int &n = number;
   n = n * n * n;
   cout << number << endl;
}
```

Output

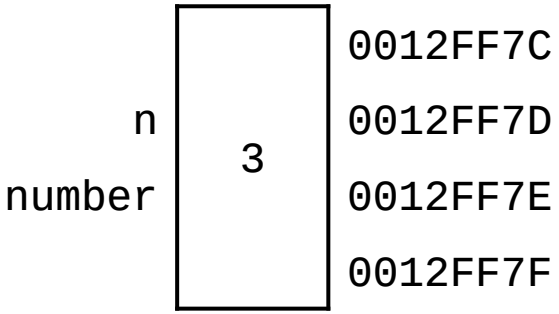





```
int main()
{
    int number = 3;
    cout << number << endl;
    int &n = number;
    n = n * n * n;
    cout << number << endl;
}
```

Output

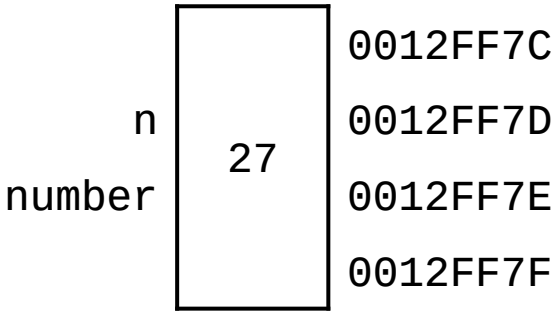
```
3
```




```
int main()
{
    int number = 3;
    cout << number << endl;
    int &n = number;
    n = n * n * n;
    cout << number << endl;
}
```

Output

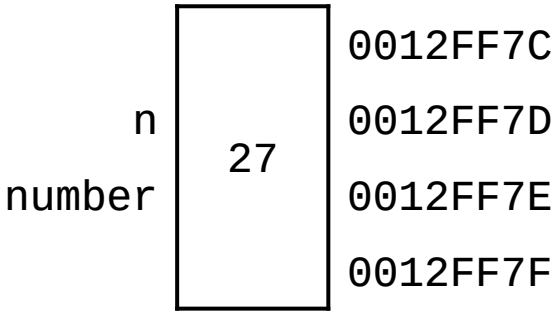
```
3
```




```
int main()
{
    int number = 3;
    cout << number << endl;
    int &n = number;
    n = n * n * n;
    cout << number << endl;
}
```

Output

```
3
```

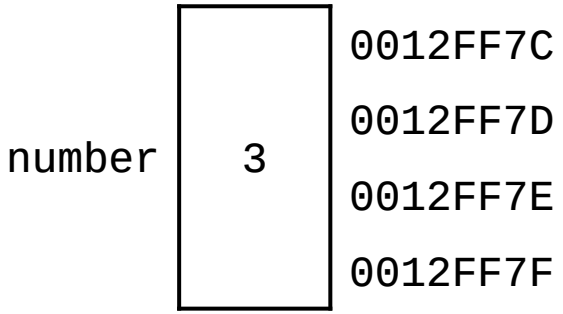



```
int main()
{
    int number = 3;
    cout << number << endl;
    int &n = number;
    n = n * n * n;
    cout << number << endl;
}
```

Output

```
3
27
```

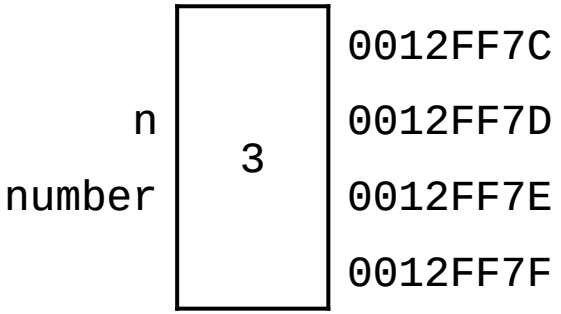
Comparison




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   int &n = number;
   n = n * n * n;
   cout << number << endl;
}
```

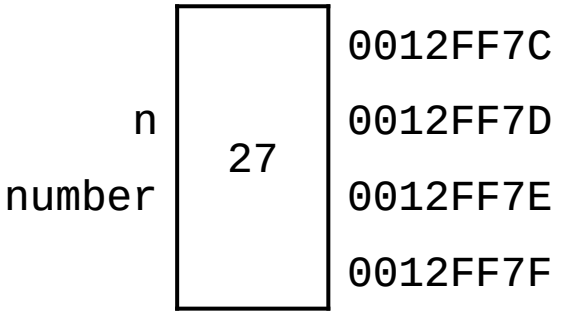
Comparison




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   int &n = number;
   n = n * n * n;
   cout << number << endl;
}
```

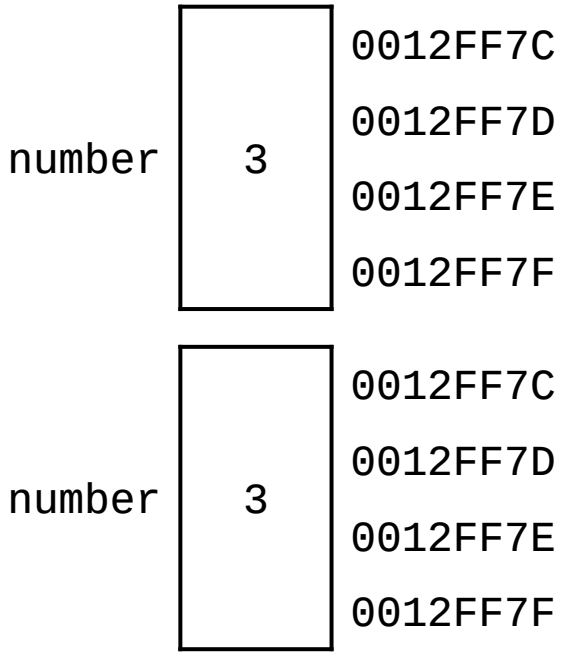
Comparison




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   int &n = number;
   n = n * n * n;
   cout << number << endl;
}
```

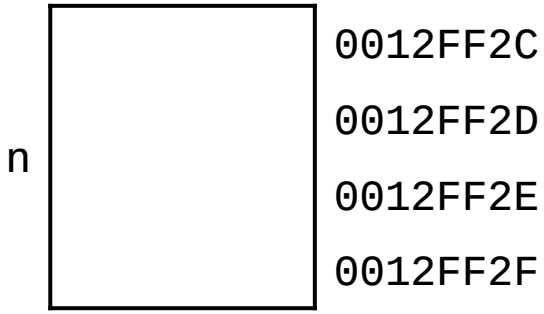
Comparison

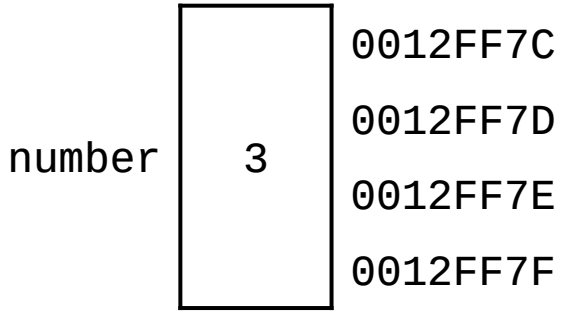


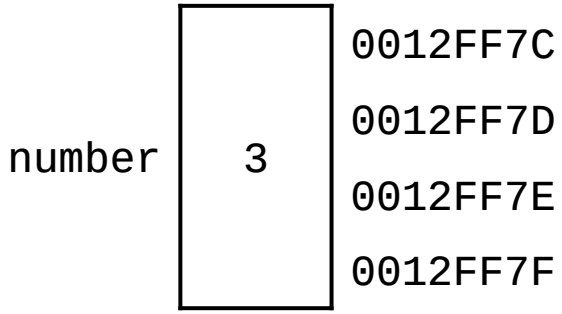

```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Comparison



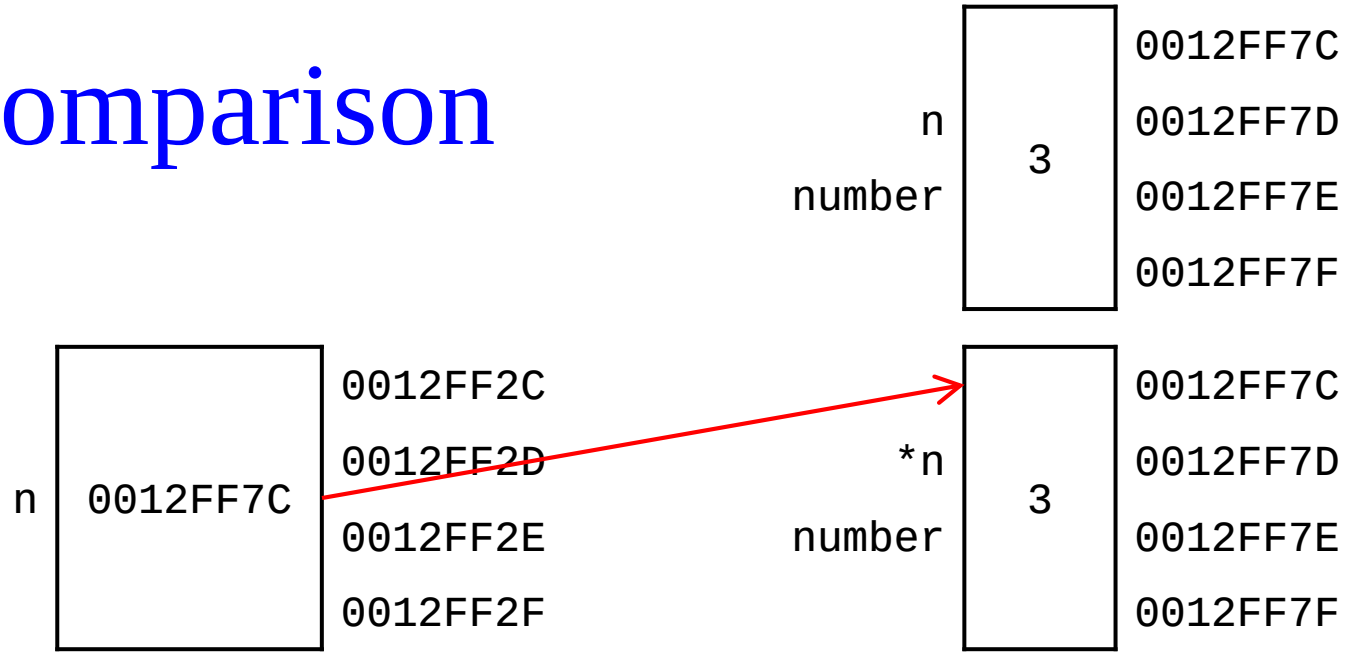





```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

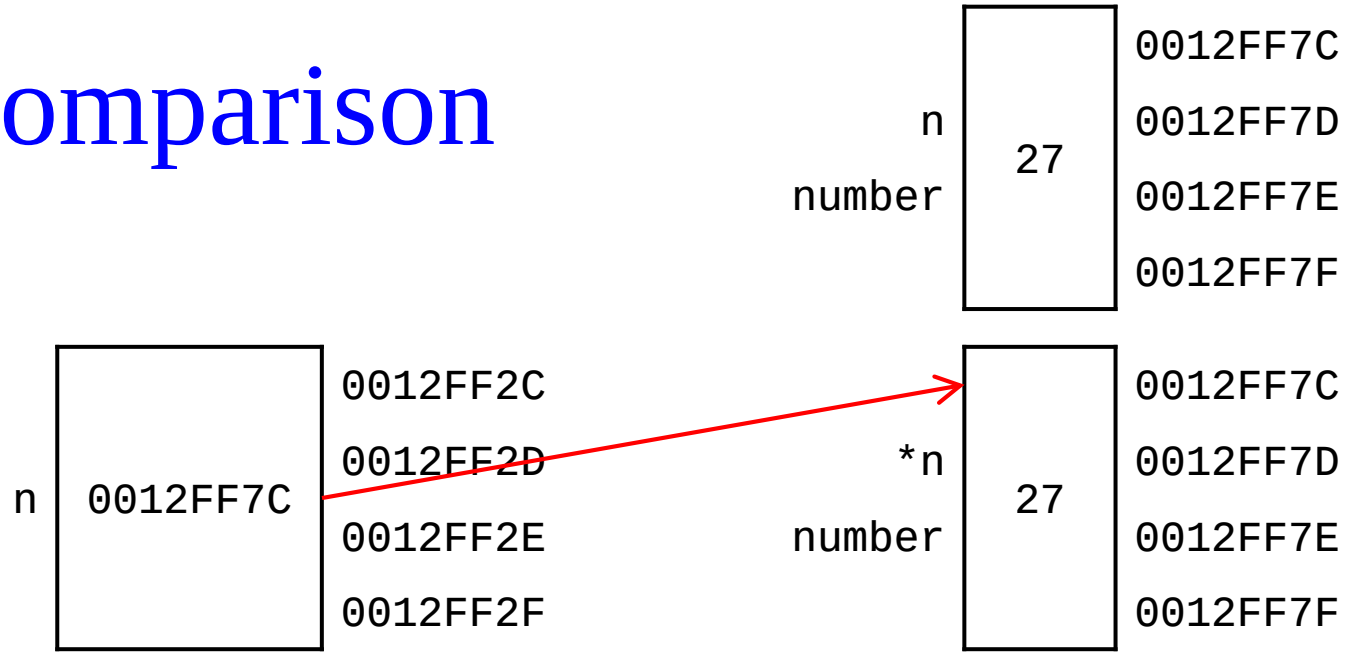
Comparison




```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Comparison

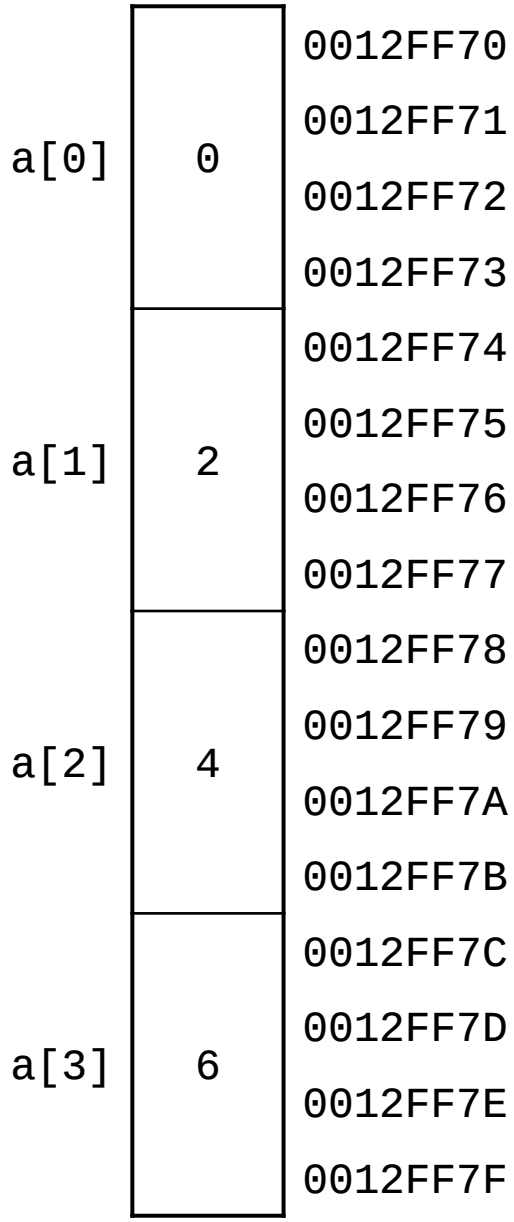



```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( &number );
   cout << number << endl;
}

void cube( int *n )
{
   *n = *n * *n * *n;
}
```

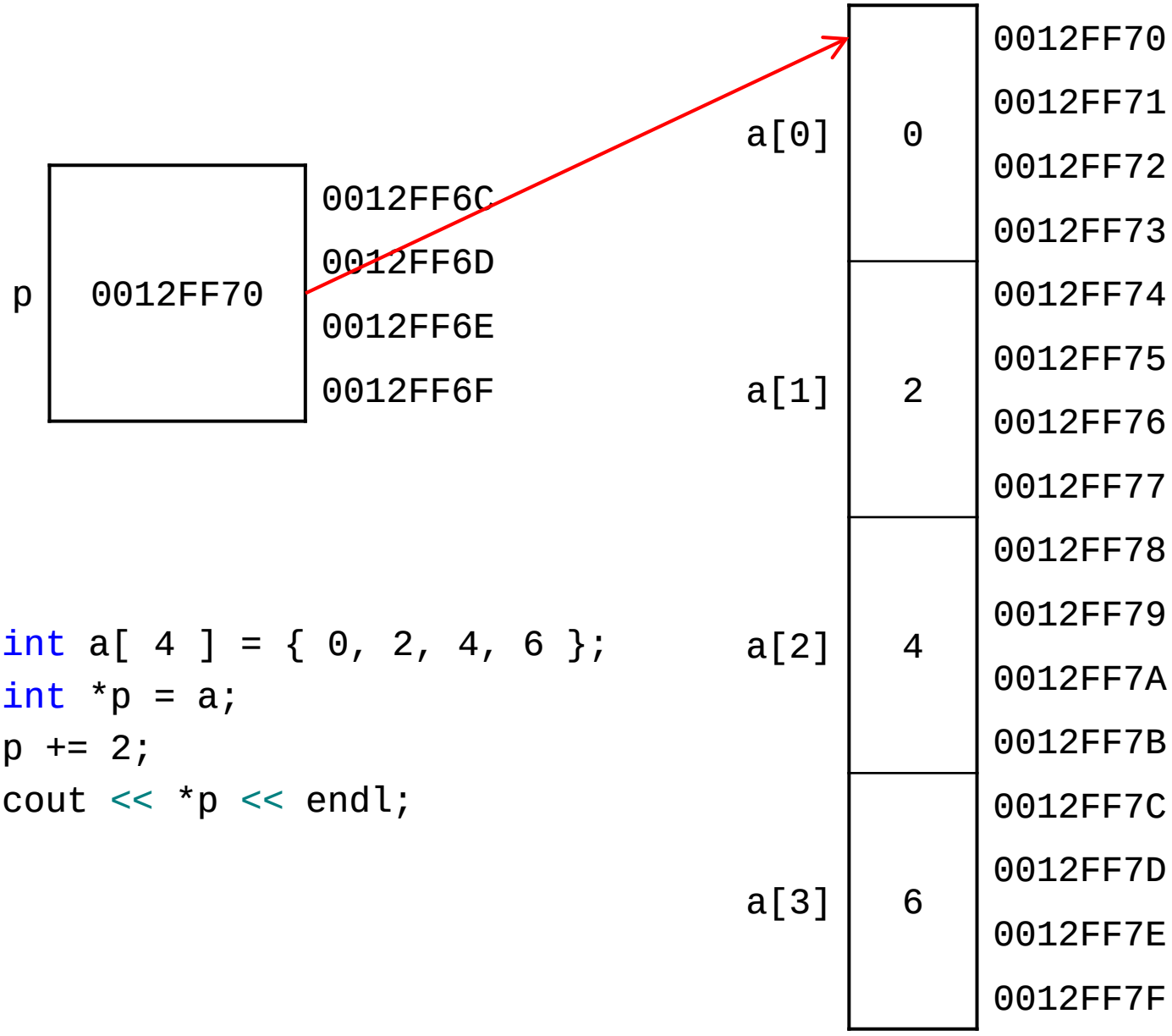
```
int main()
{
   int number = 3;
   cout << number << endl;
   cube( number );
   cout << number << endl;
}

void cube( int &n )
{
   n = n * n * n;
}
```

Pointer Arithmetic

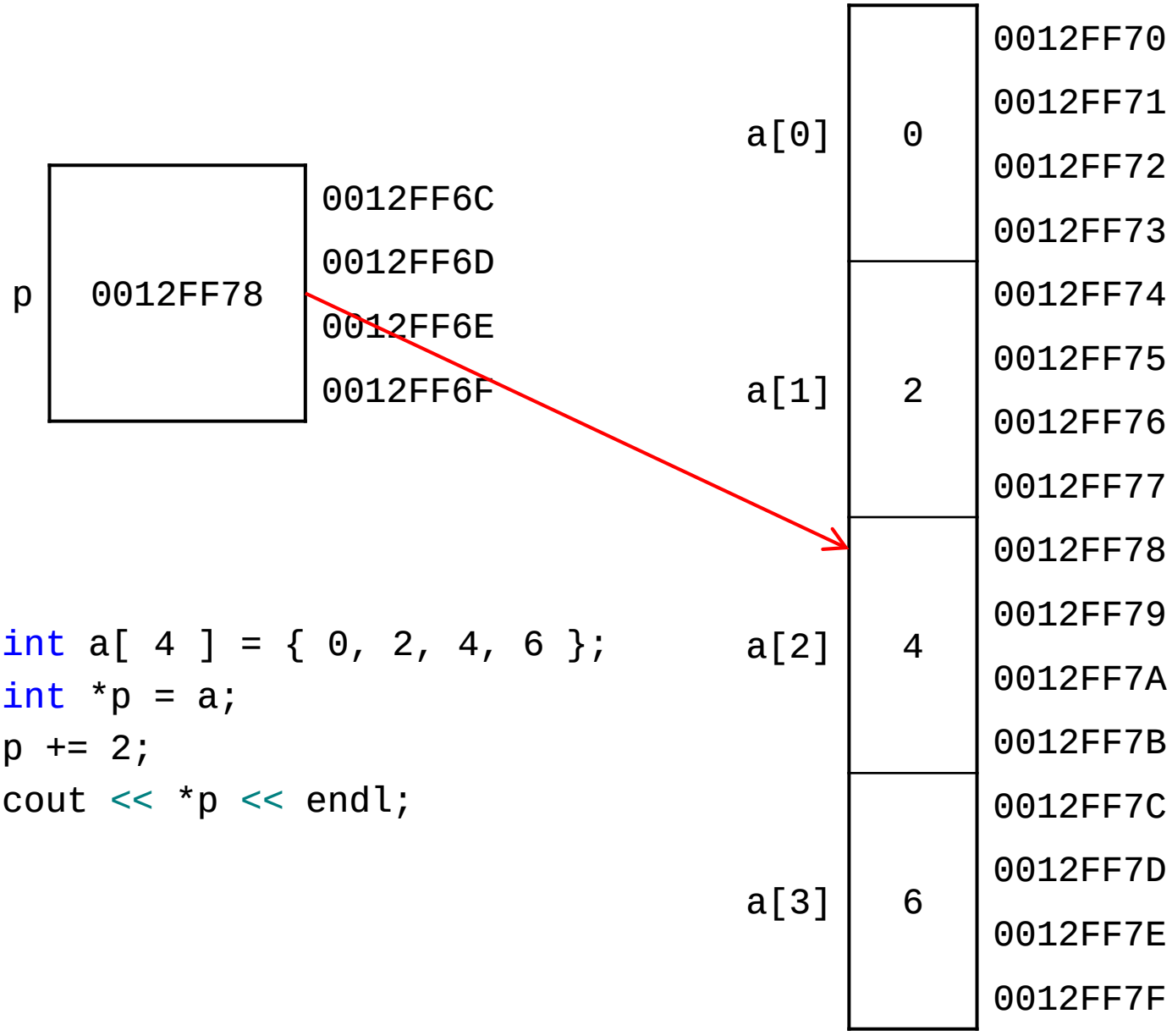
- increment (++)
- decremented (--)
- an integer may be added to a pointer (+ or +=)
- an integer may be subtracted from a pointer (- or -=)
- one pointer may be subtracted from another of the same type



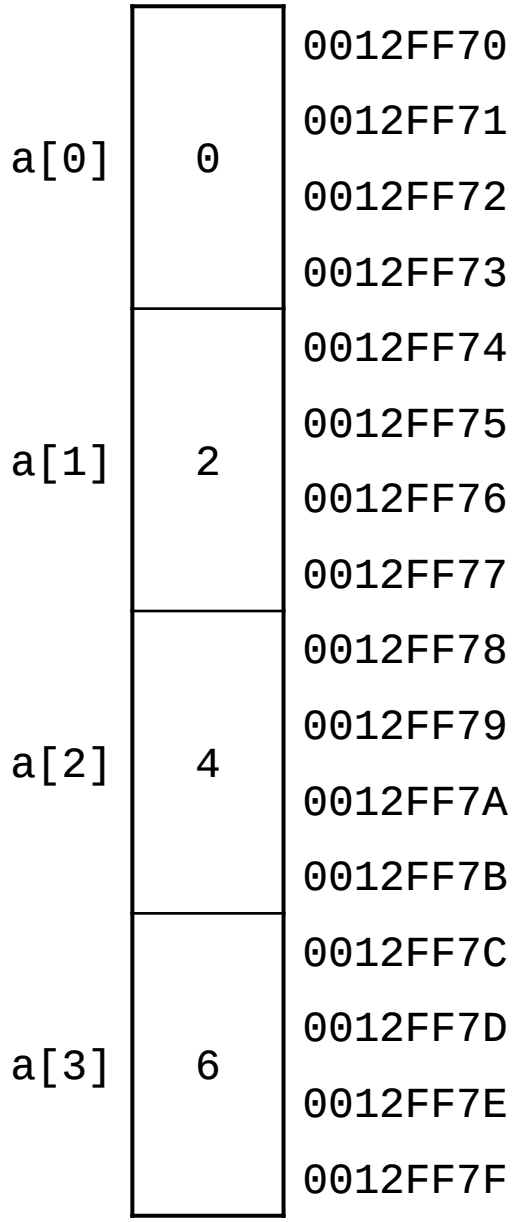

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = a;
p += 2;
cout << *p << endl;
```



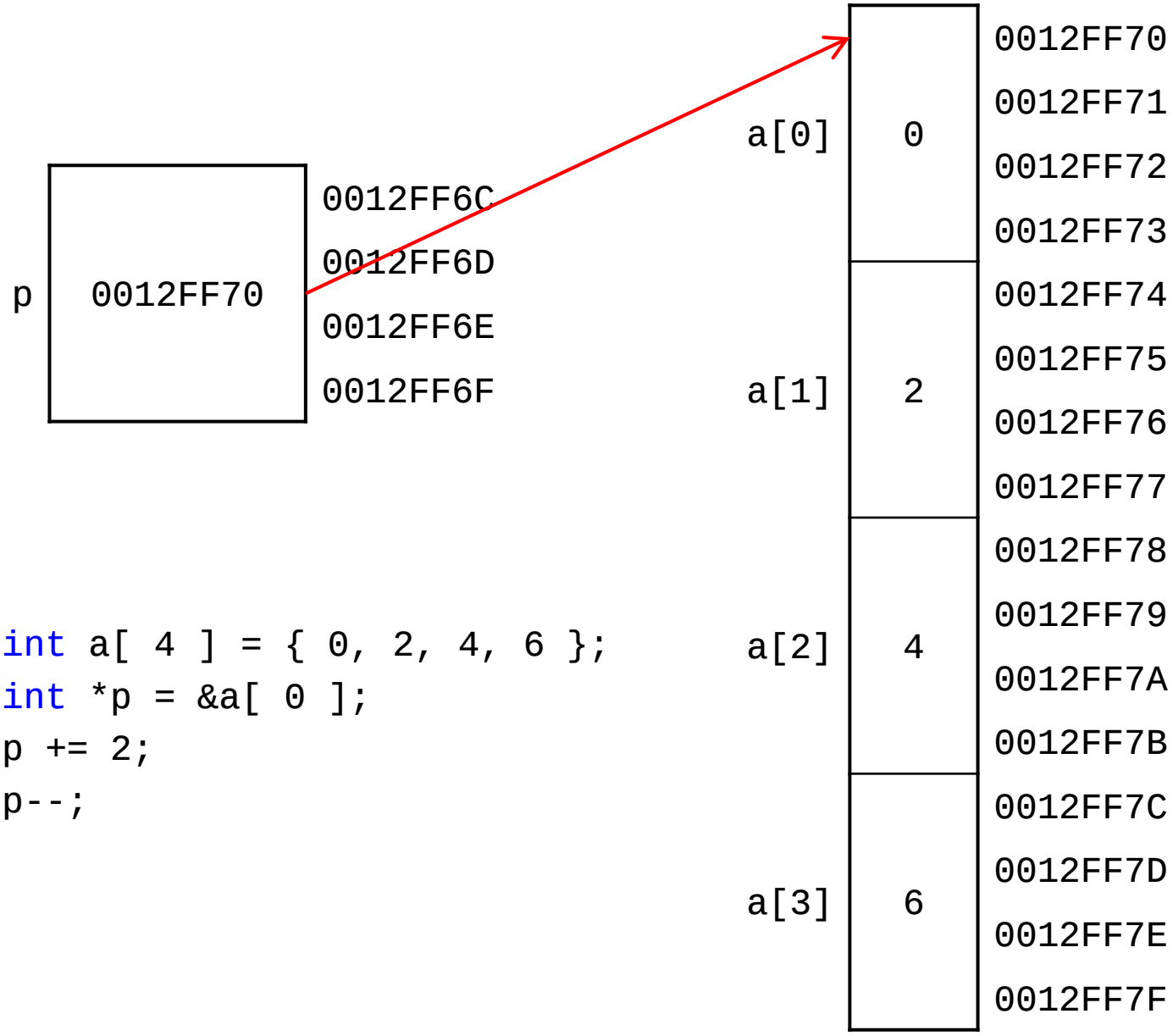

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = a;
p += 2;
cout << *p << endl;
```



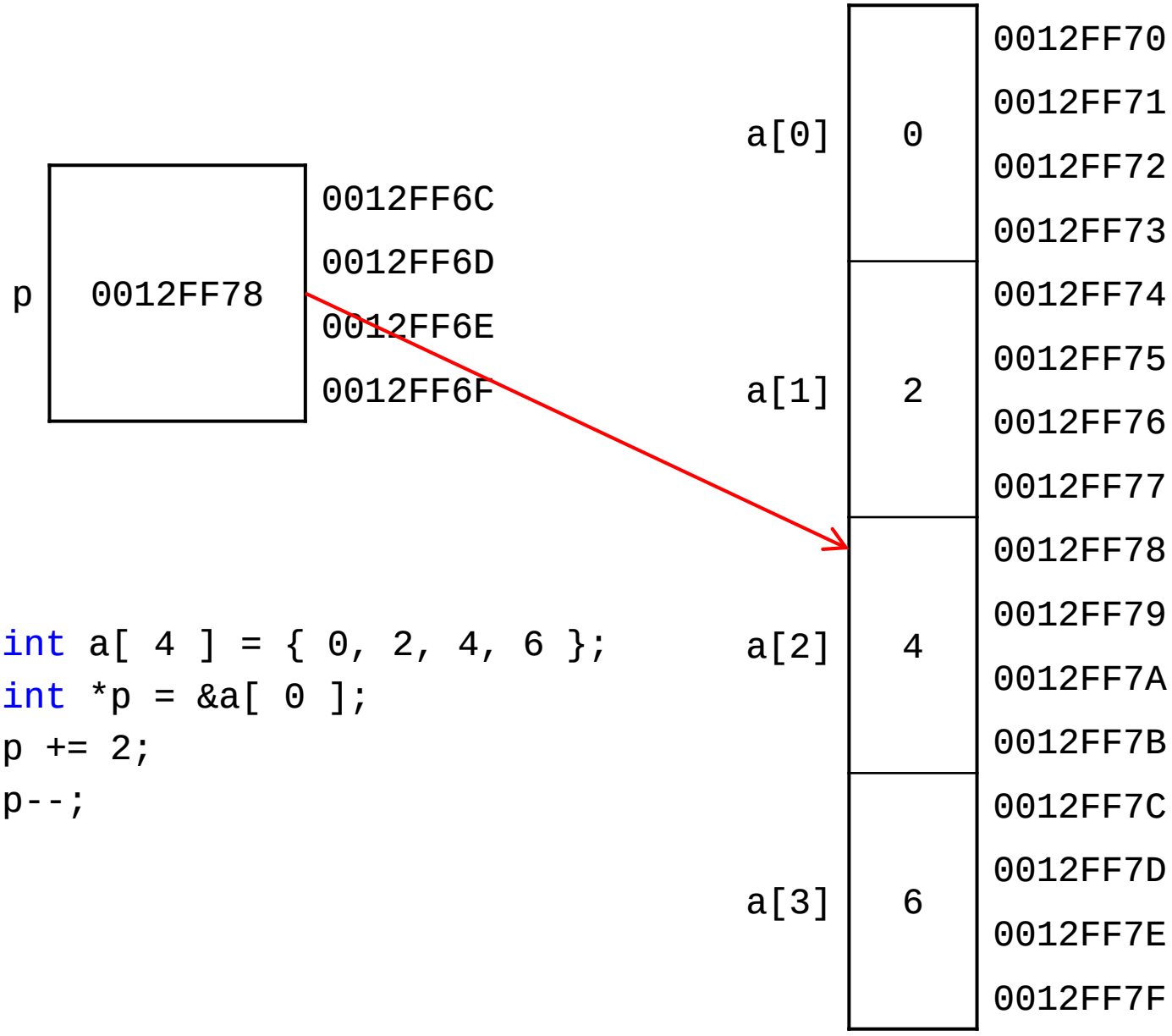

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = a;
p += 2;
cout << *p << endl;
```



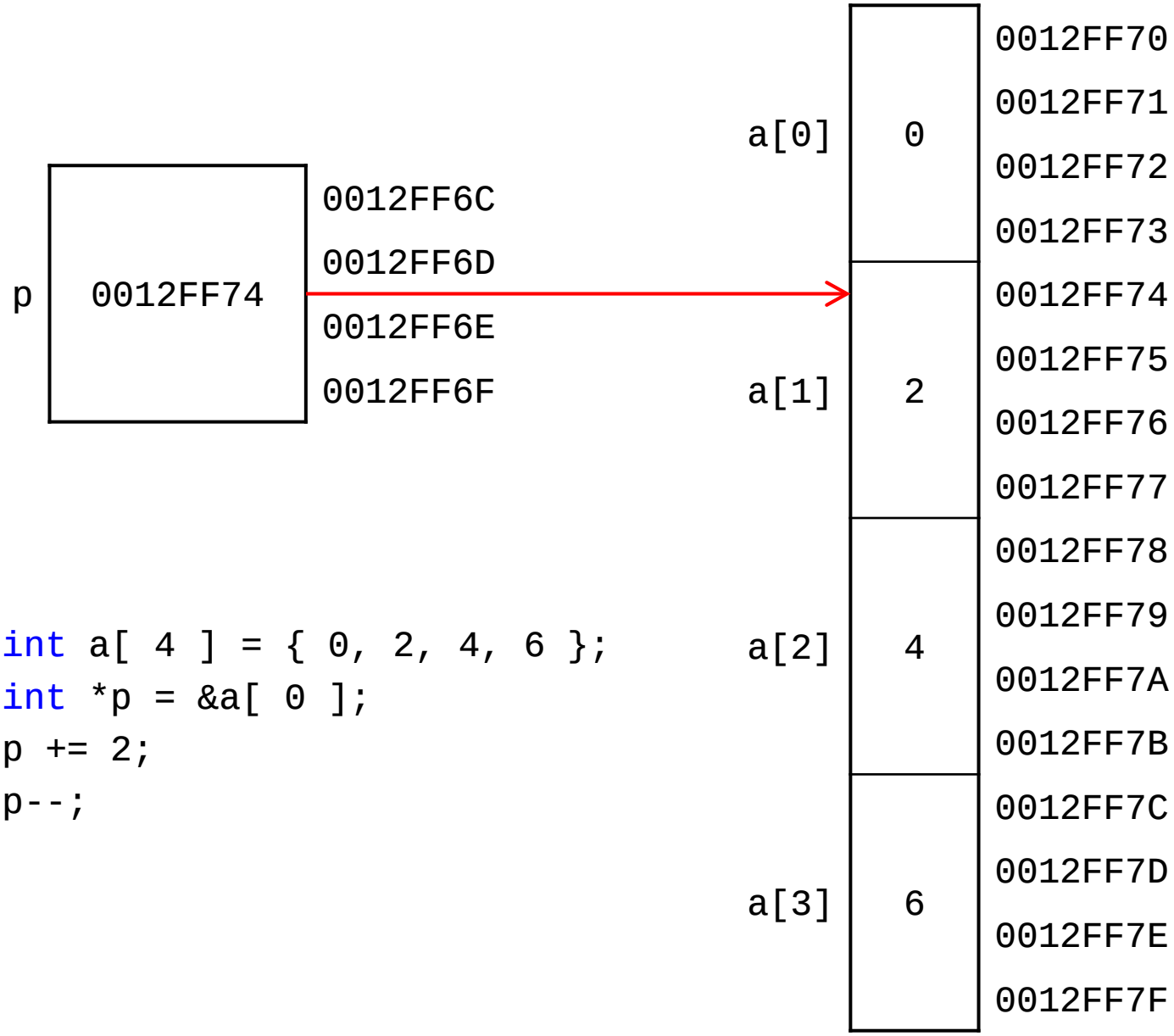

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = &a[ 0 ];
p += 2;
p--;
```



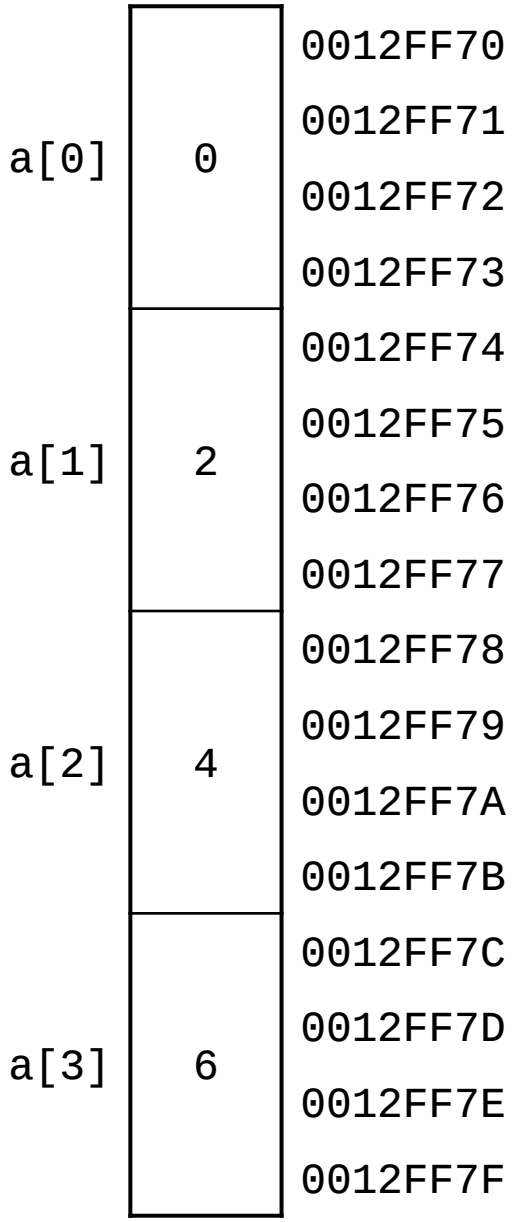

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = &a[ 0 ];
p += 2;
p--;
```



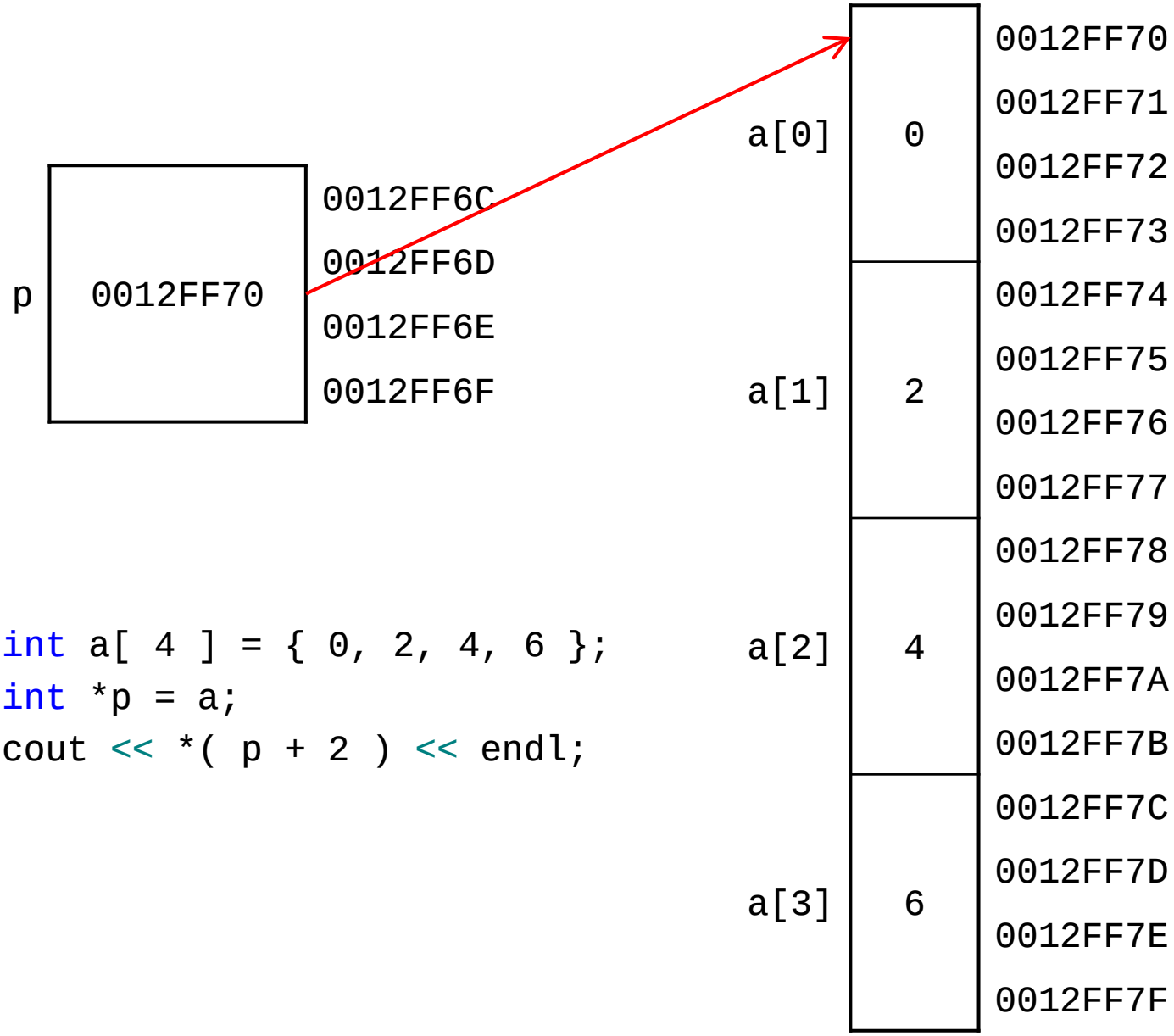

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = &a[ 0 ];
p += 2;
p--;
```



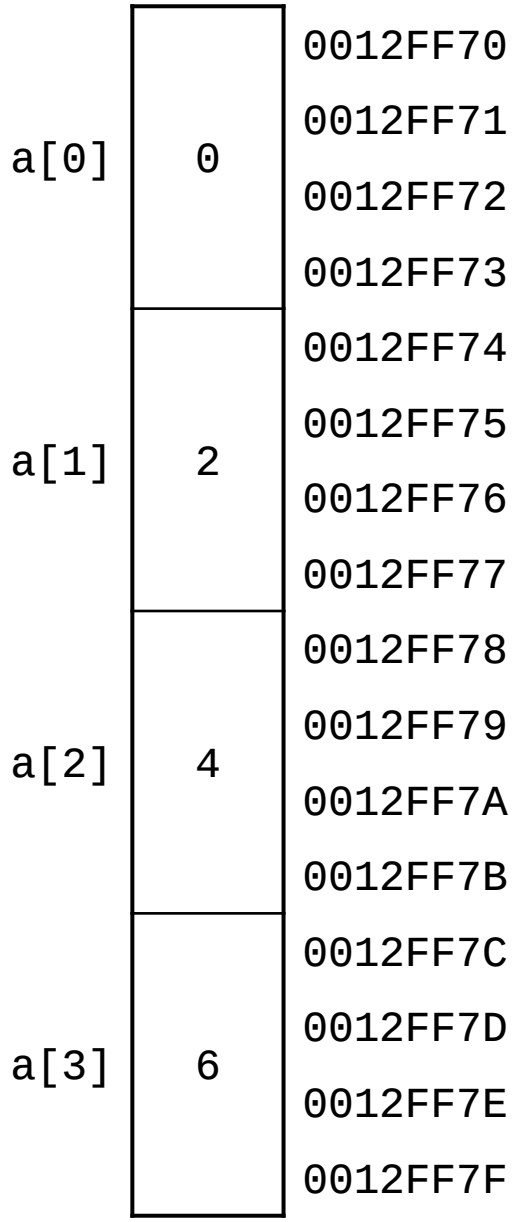

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = &a[ 0 ];
p += 2;
p--;
```



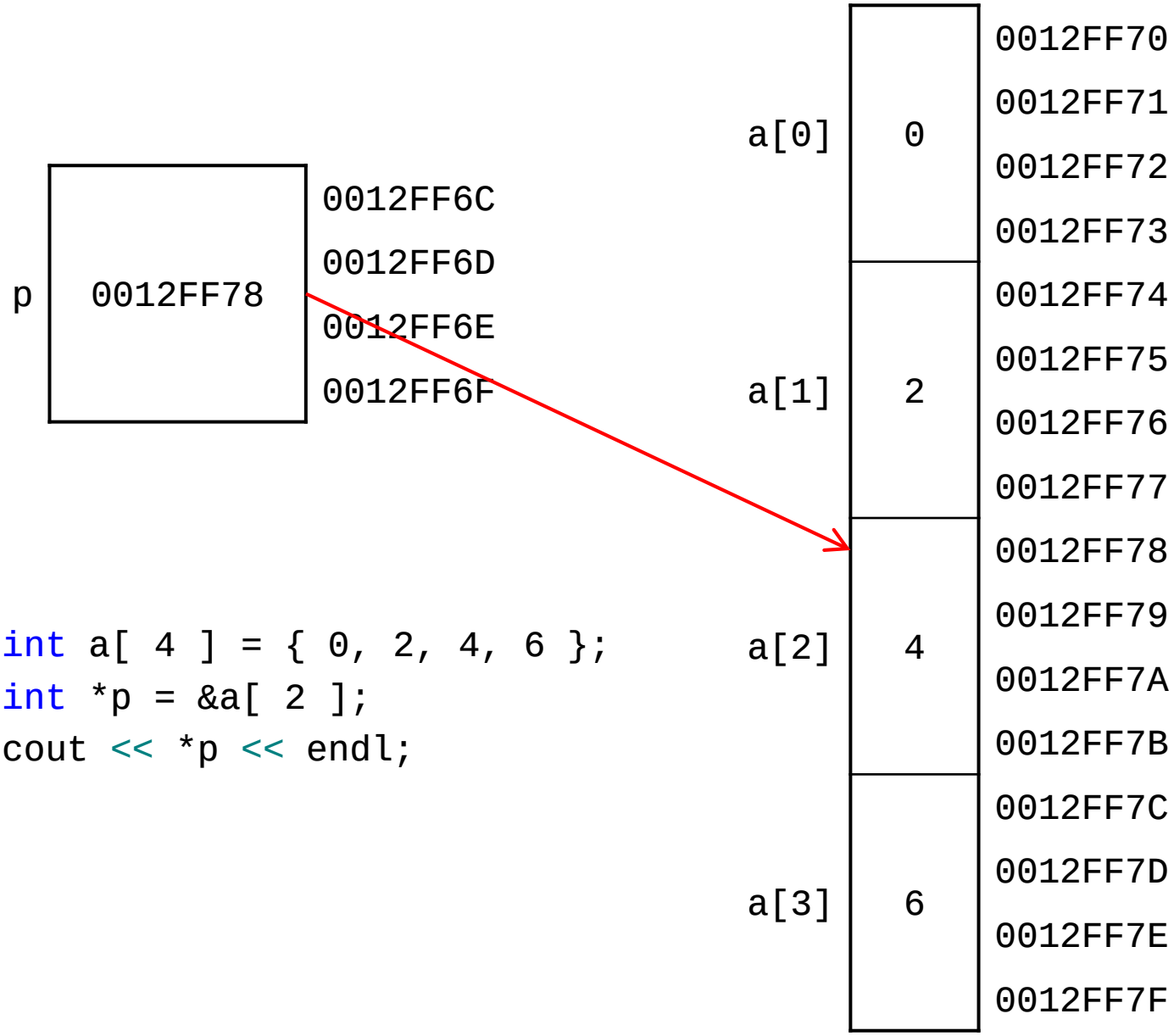

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = a;
cout << *( p + 2 ) << endl;
```



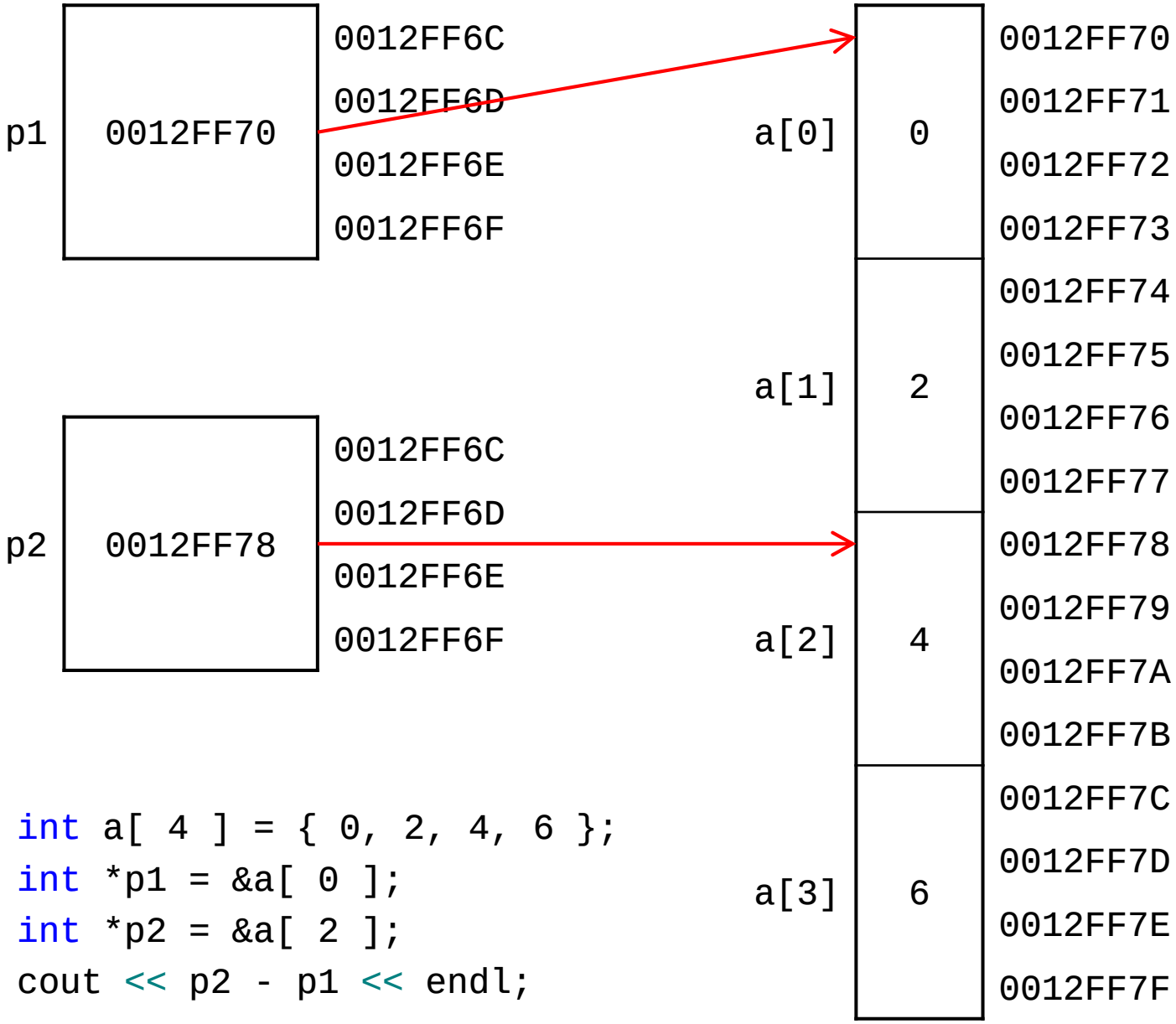

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = a;
cout << *( p + 2 ) << endl;
```




```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = &a[ 2 ];
cout << *p << endl;
```



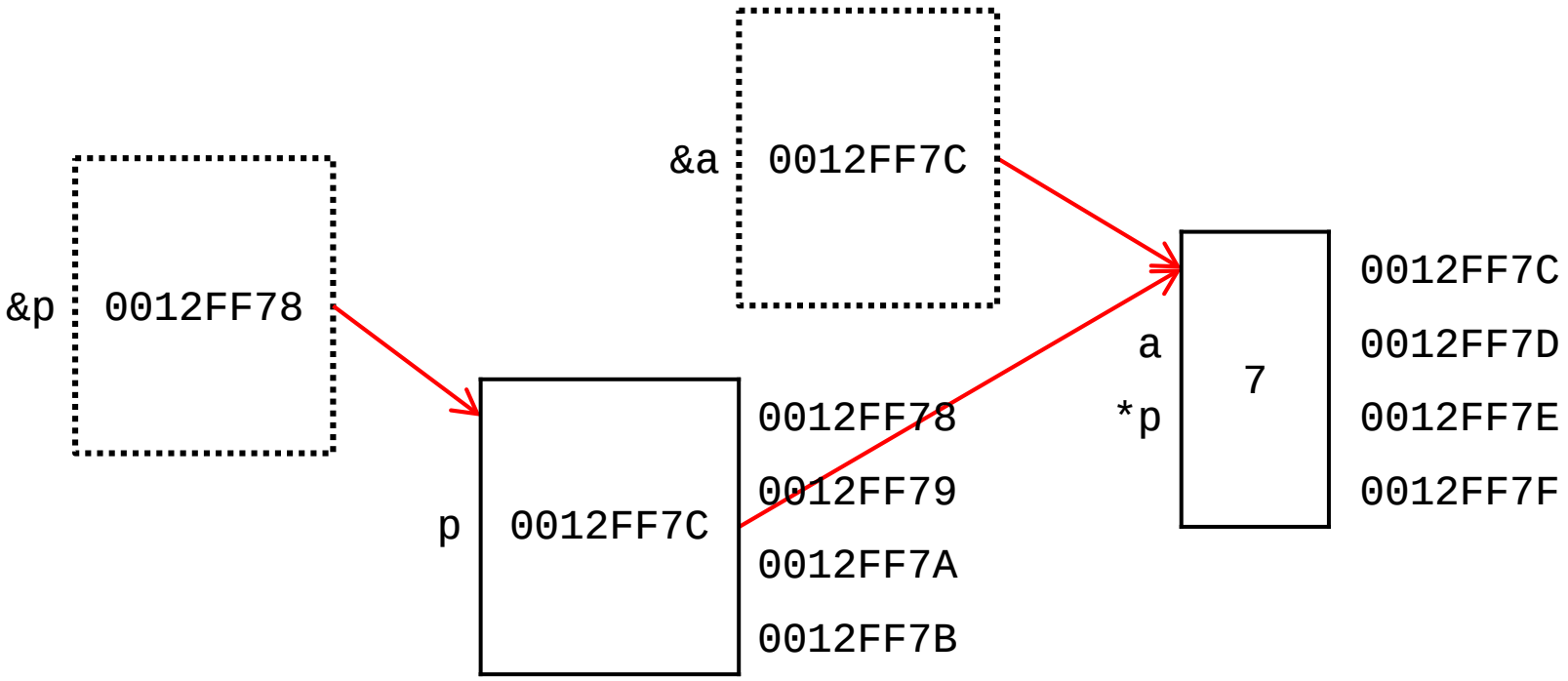

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p = &a[ 2 ];
cout << *p << endl;
```

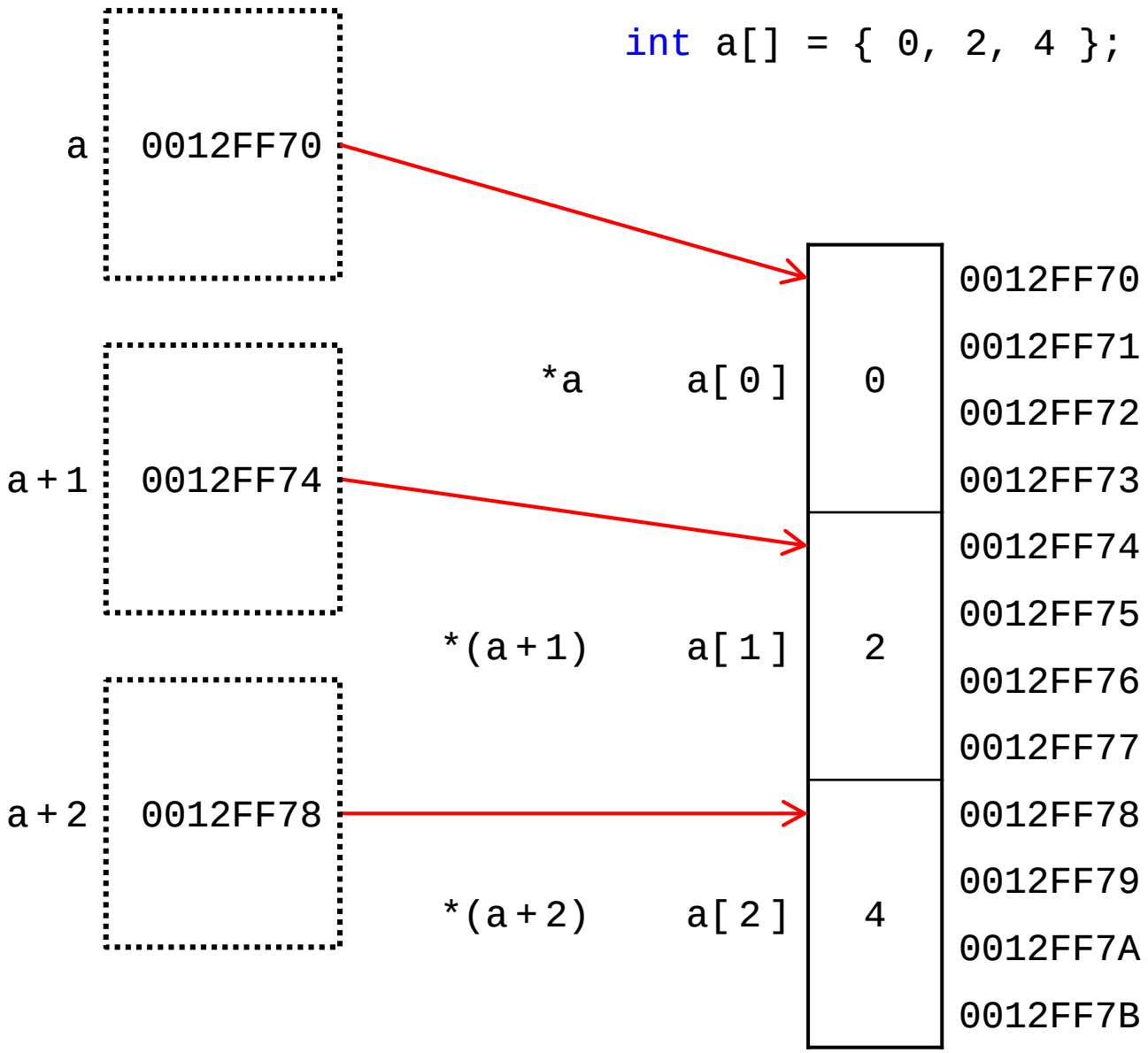


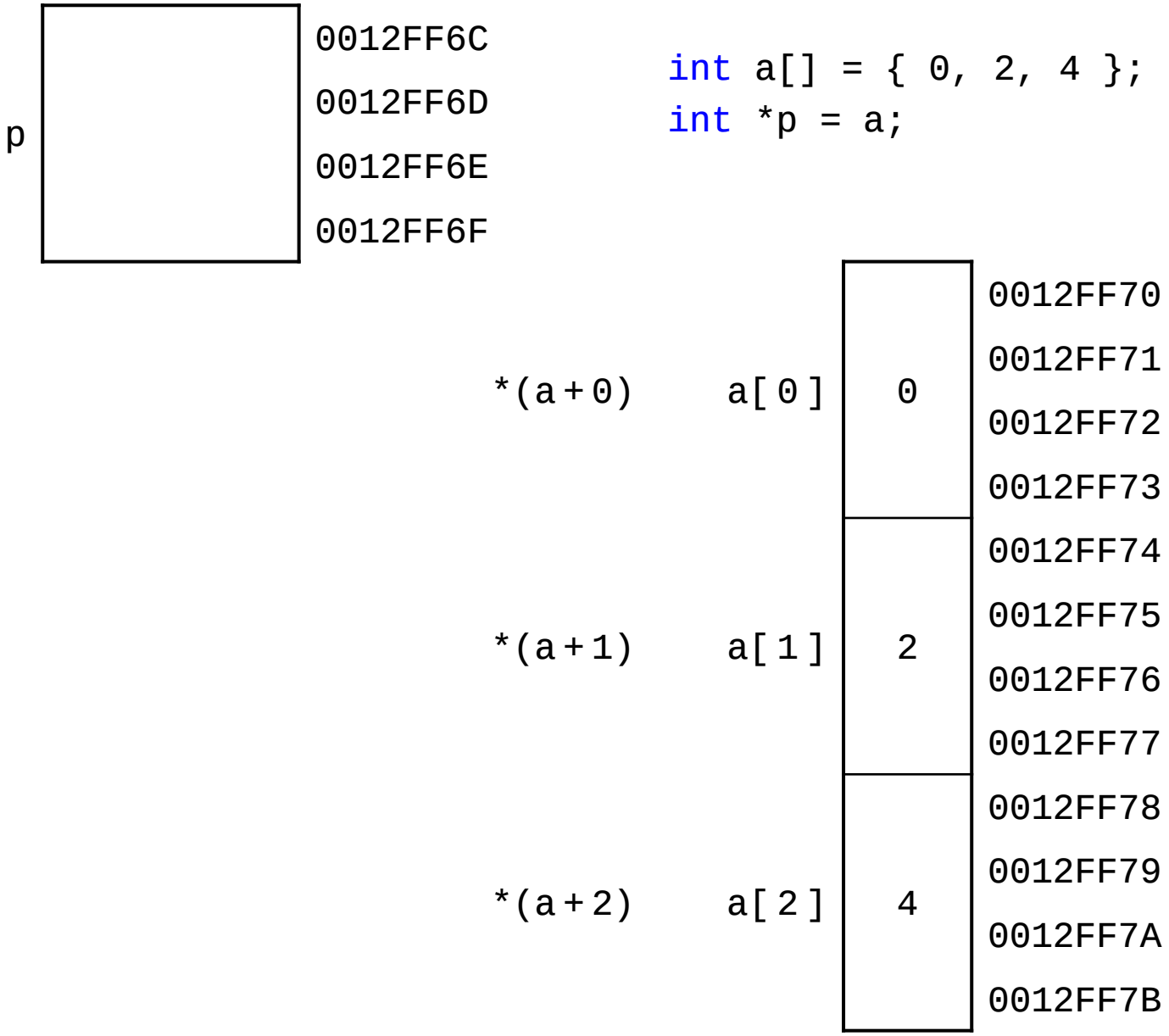

```
int a[ 4 ] = { 0, 2, 4, 6 };
int *p1 = &a[ 0 ];
int *p2 = &a[ 2 ];
cout << p2 - p1 << endl;
```

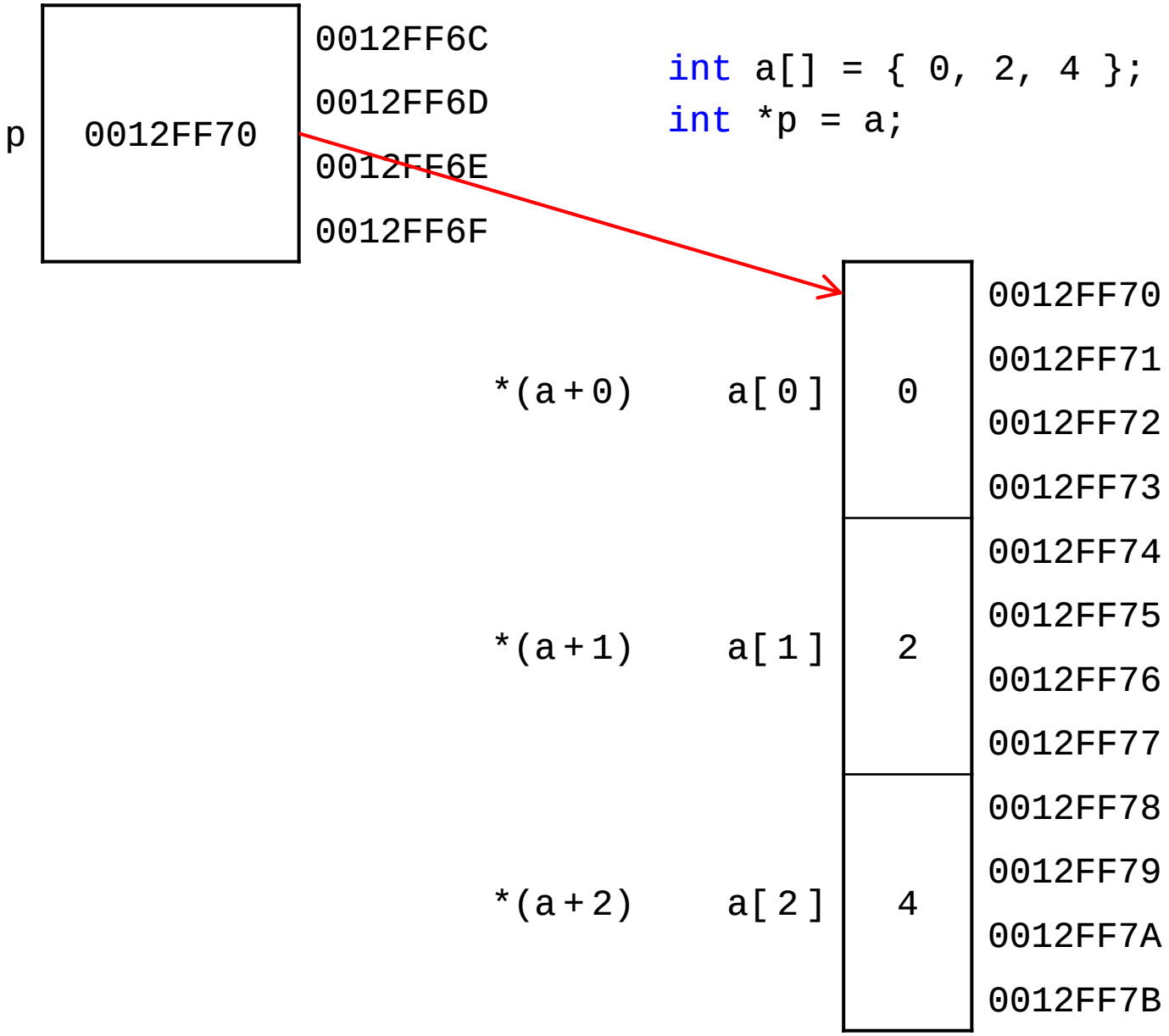
Relationship Between Pointers and Arrays

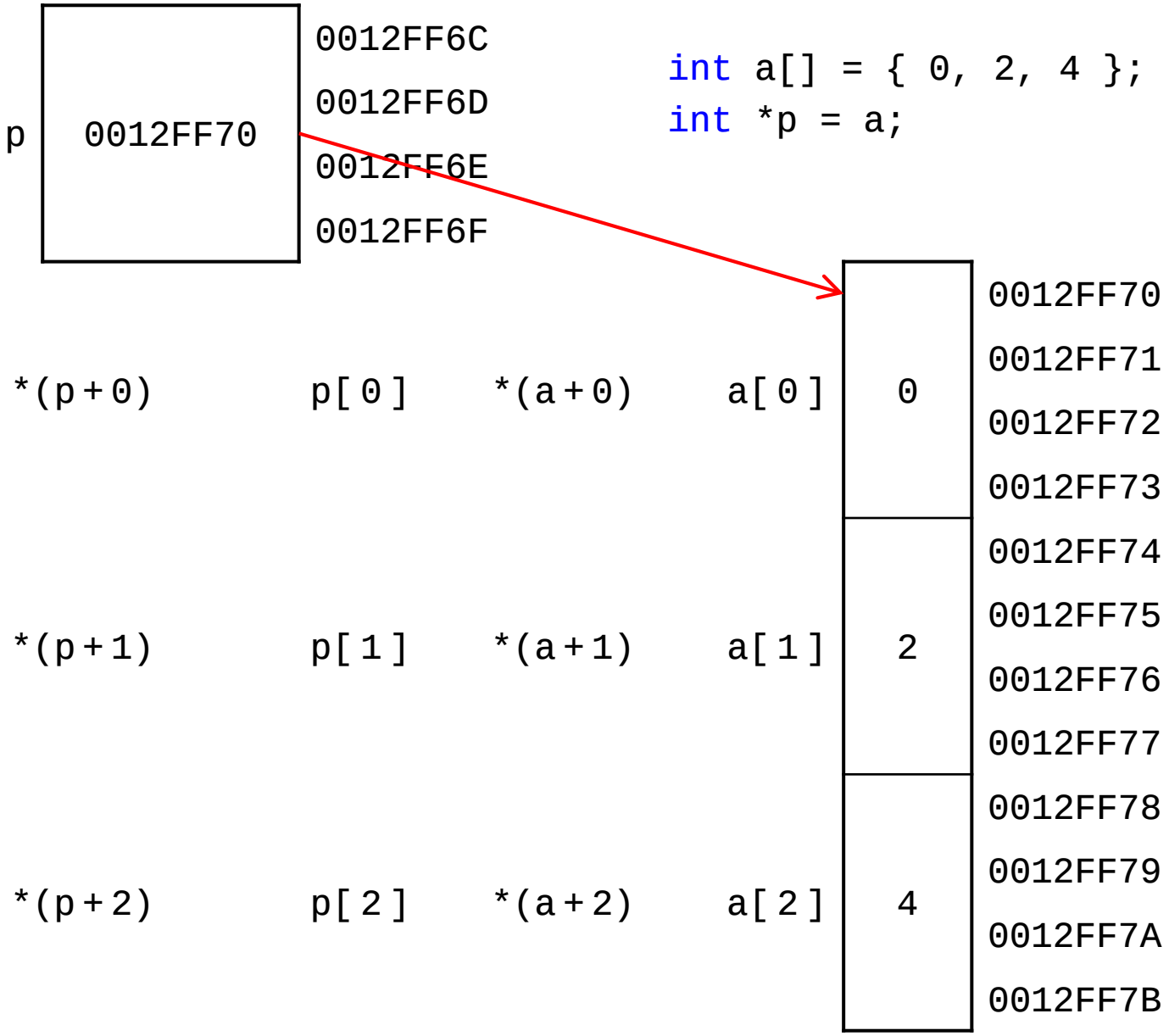
```
int a = 7;
int *p = &a;
cout << &*p << " " << *&p << endl;
```

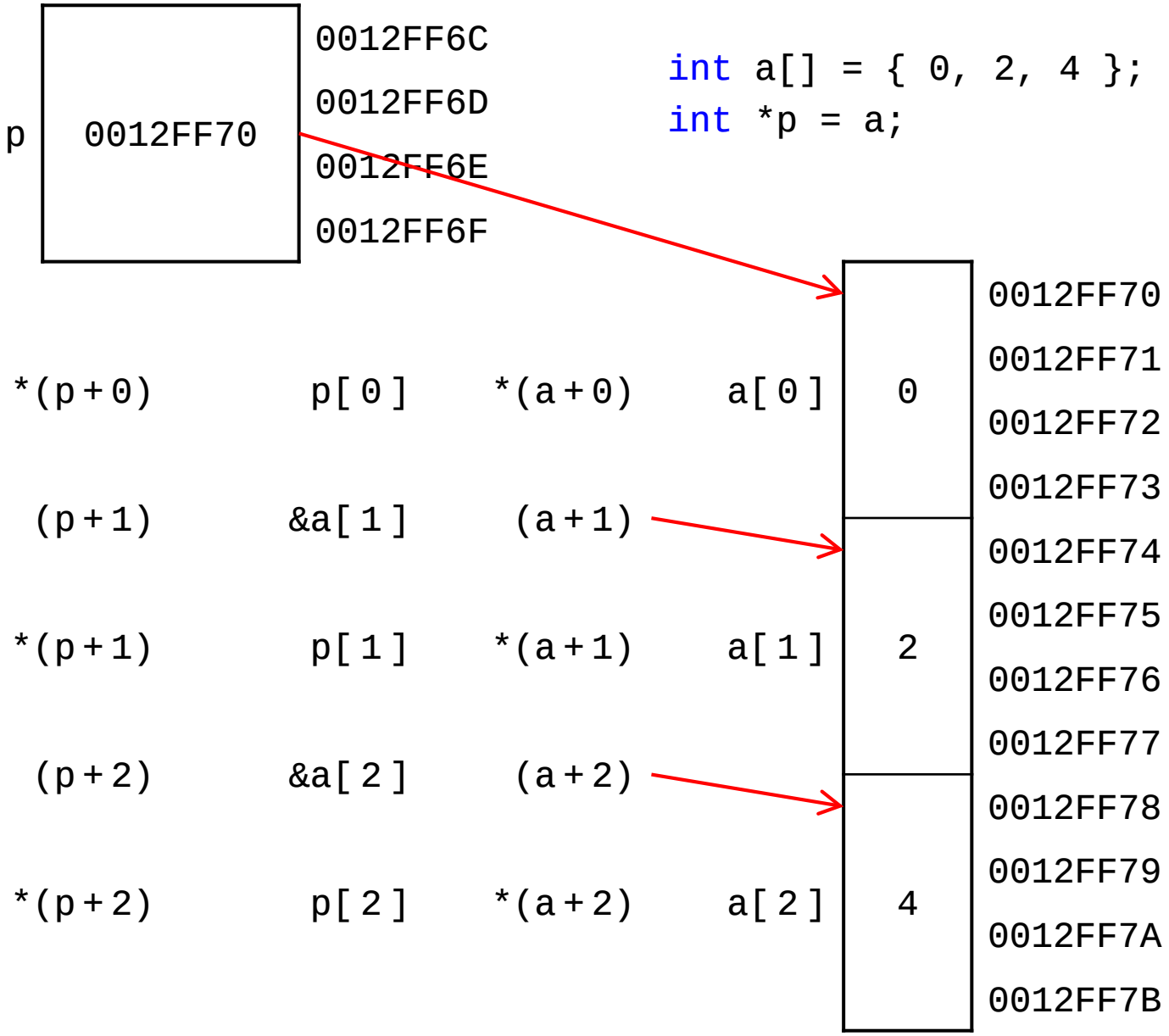


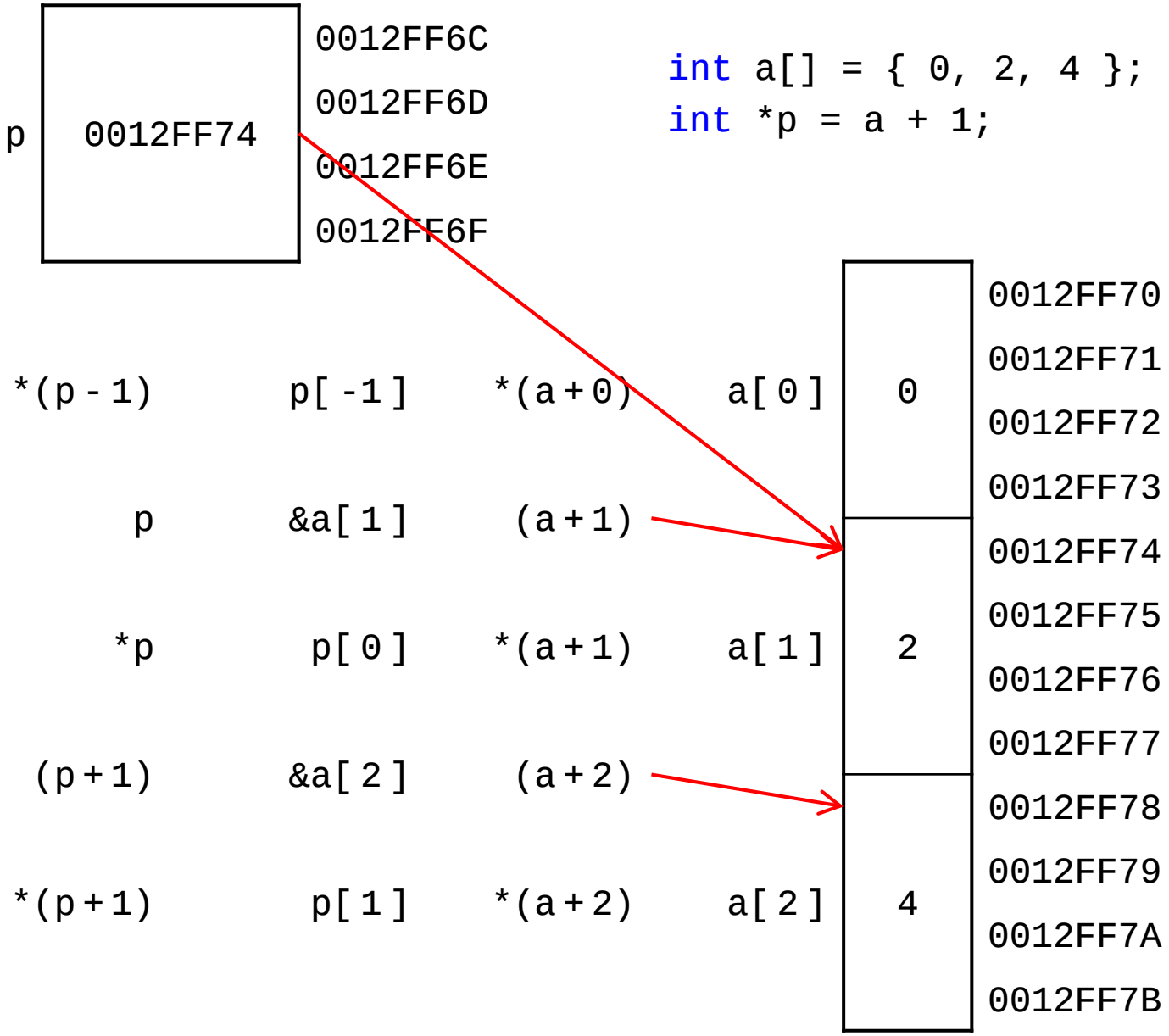

int a[] = { 0, 2, 4 };
a
0012FF70
a+1
0012FF74
a+2
0012FF78
*a
a[0]
*(a+1)
a[1]
*(a+2)
a[2]
0
2
4
0012FF70
0012FF71
0012FF72
0012FF73
0012FF74
0012FF75
0012FF76
0012FF77
0012FF78
0012FF79
0012FF7A
0012FF7B

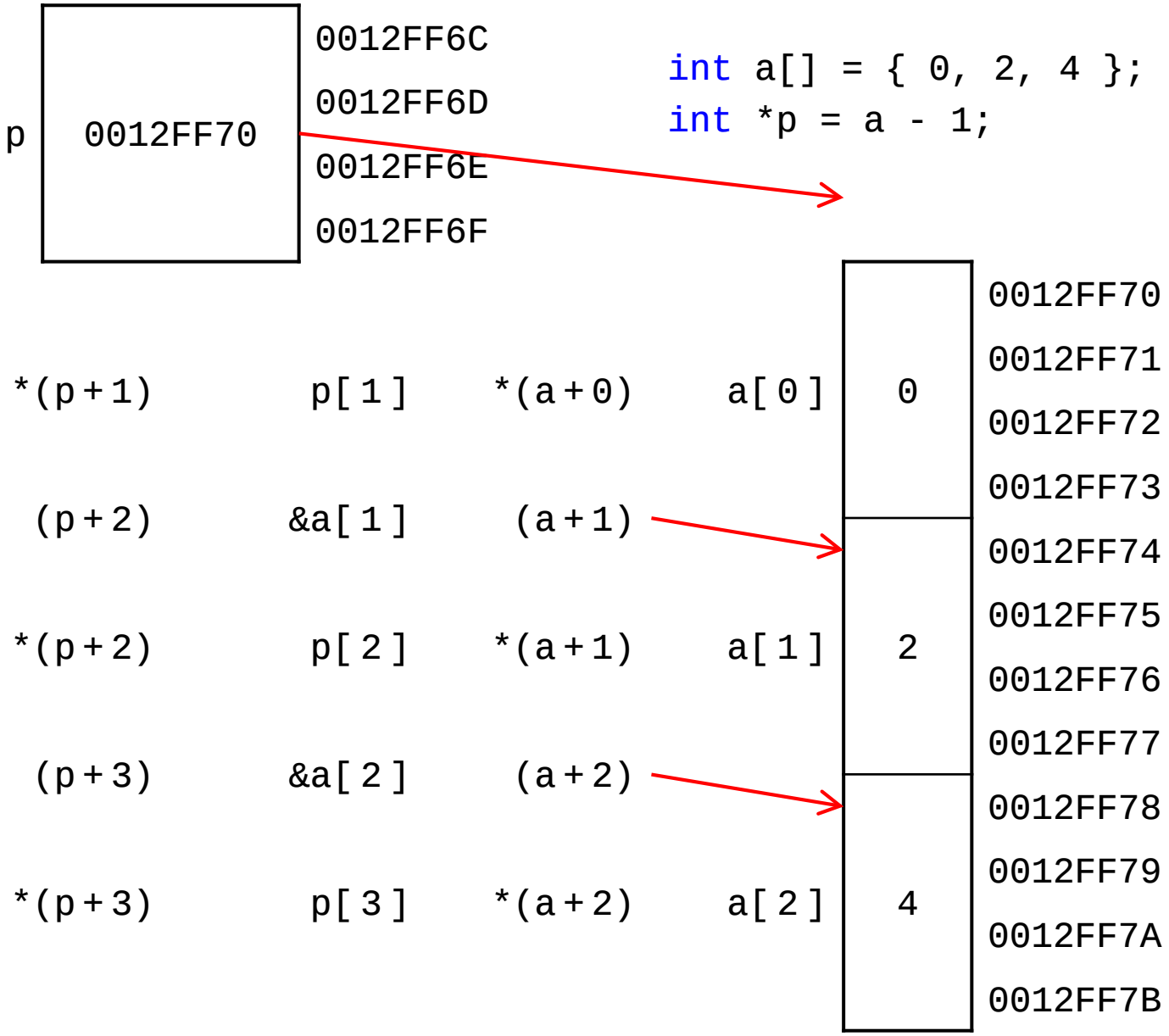

p
0012FF6C
0012FF6D
0012FF6E
0012FF6F
int a[] = { 0, 2, 4 };
int *p = a;
*(a+0)
a[0]
0
0012FF70
0012FF71
0012FF72
0012FF73
*(a+1)
a[1]
2
0012FF74
0012FF75
0012FF76
0012FF77
*(a+2)
a[2]
4
0012FF78
0012FF79
0012FF7A
0012FF7B

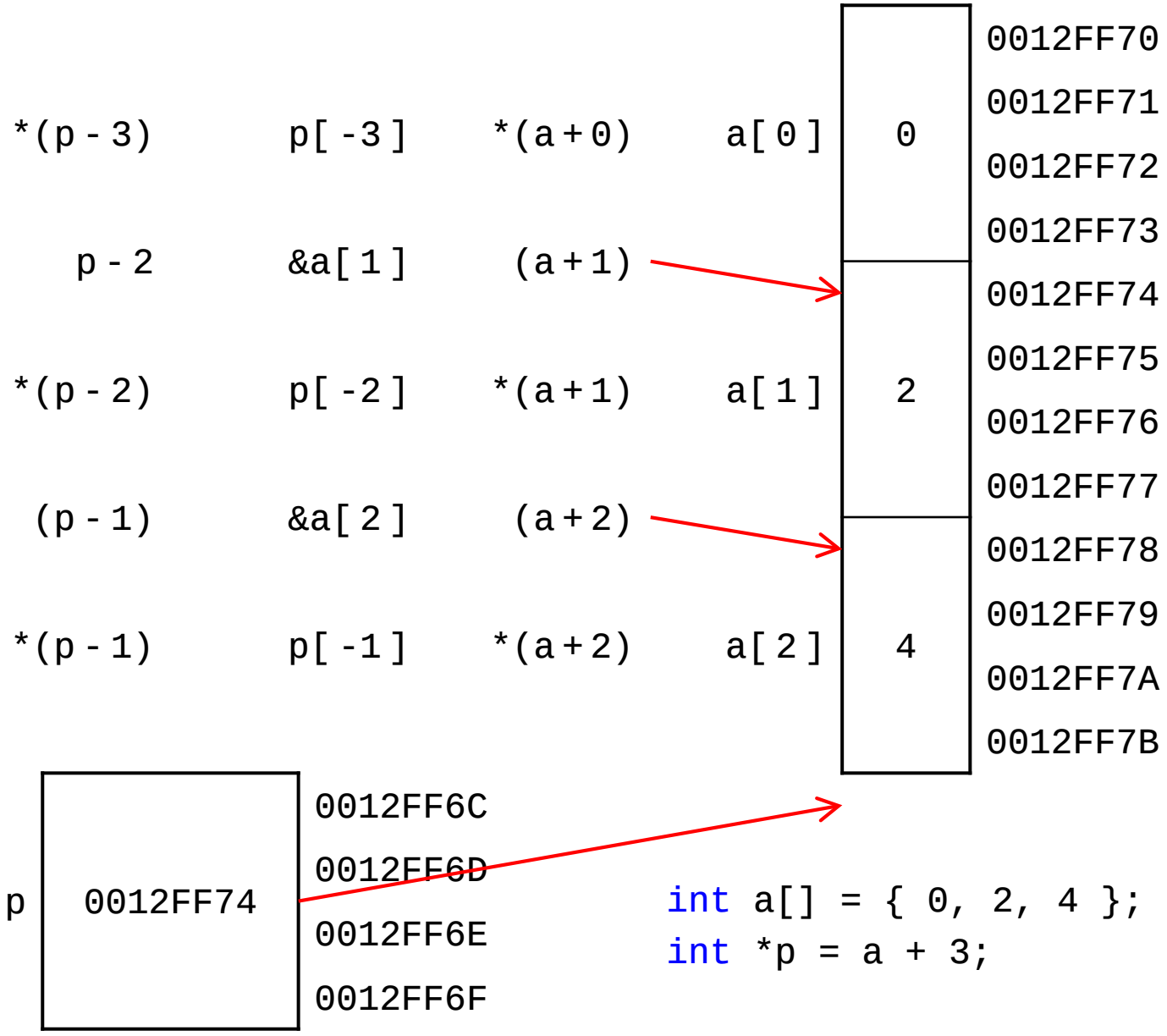

p
0012FF70
0012FF6C
0012FF6D
0012FF6E
0012FF6F
int a[] = { 0, 2, 4 };
int *p = a;
*(a+0)
a[0]
0
*(a+1)
a[1]
2
*(a+2)
a[2]
4
0012FF70
0012FF71
0012FF72
0012FF73
0012FF74
0012FF75
0012FF76
0012FF77
0012FF78
0012FF79
0012FF7A
0012FF7B


p
0012FF70
0012FF6C
0012FF6D
0012FF6E
0012FF6F
int a[] = { 0, 2, 4 };
int *p = a;
0012FF70
0012FF71
0012FF72
0012FF73
0012FF74
0012FF75
0012FF76
0012FF77
0012FF78
0012FF79
0012FF7A
0012FF7B
0
2
4
*(p+0)
p[0]
*(a+0)
a[0]
*(p+1)
p[1]
*(a+1)
a[1]
*(p+2)
p[2]
*(a+2)
a[2]


p
0012FF70
0012FF6C
0012FF6D
0012FF6E
0012FF6F
int a[] = { 0, 2, 4 };
int *p = a;
0012FF70
0012FF71
0012FF72
0012FF73
0012FF74
0012FF75
0012FF76
0012FF77
0012FF78
0012FF79
0012FF7A
0012FF7B
0
2
4
*(p+0) p[0] *(a+0) a[0]
(p+1) &a[1] (a+1)
*(p+1) p[1] *(a+1) a[1]
(p+2) &a[2] (a+2)
*(p+2) p[2] *(a+2) a[2]


p
0012FF74
0012FF6C
0012FF6D
0012FF6E
0012FF6F
int a[] = { 0, 2, 4 };
int *p = a + 1;
*(p - 1)
p[-1]
*(a + 0)
a[0]
0
p
&a[1]
(a + 1)
*p
p[0]
*(a + 1)
a[1]
2
(p + 1)
&a[2]
(a + 2)
*(p + 1)
p[1]
*(a + 2)
a[2]
4
0012FF70
0012FF71
0012FF72
0012FF73
0012FF74
0012FF75
0012FF76
0012FF77
0012FF78
0012FF79
0012FF7A
0012FF7B

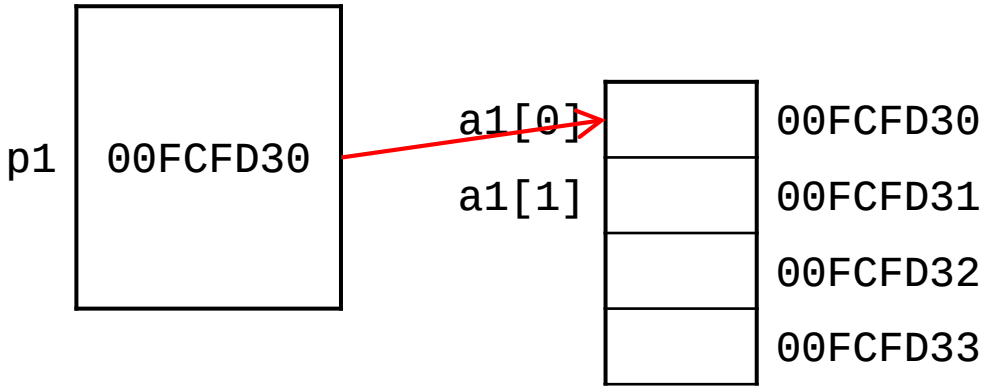

p
0012FF70
0012FF6C
0012FF6D
0012FF6E
0012FF6F
int a[] = { 0, 2, 4 };
int *p = a - 1;
0012FF70
0012FF71
0012FF72
0012FF73
0012FF74
0012FF75
0012FF76
0012FF77
0012FF78
0012FF79
0012FF7A
0012FF7B
0
2
4
*(p+1) p[1] *(a+0) a[0]
(p+2) &a[1] (a+1)
*(p+2) p[2] *(a+1) a[1]
(p+3) &a[2] (a+2)
*(p+3) p[3] *(a+2) a[2]


*(p - 3)
p[-3]
*(a + 0)
a[0]
0
p - 2
&a[1]
(a + 1)
*(p - 2)
p[-2]
*(a + 1)
a[1]
2
(p - 1)
&a[2]
(a + 2)
*(p - 1)
p[-1]
*(a + 2)
a[2]
4
0012FF70
0012FF71
0012FF72
0012FF73
0012FF74
0012FF75
0012FF76
0012FF77
0012FF78
0012FF79
0012FF7A
0012FF7B
p
0012FF74
0012FF6C
0012FF6D
0012FF6E
0012FF6F
int a[] = { 0, 2, 4 };
int *p = a + 3;

```
int main()
{
   int a[] = { 0, 2, 4, 6 };
   for( int i = 0; i < 4; i++ )
      cout << setw( 3 ) << a[ i ];
   cout << endl;

   for( int i = 0; i < 4; i++ )
      cout << setw( 3 ) << *( a + i );
   cout << endl;

   int *p = a;
   for( int i = 0; i < 4; i++ )
      cout << setw( 3 ) << p[ i ];
   cout << endl;

   for( int i = 0; i < 4; i++ )
      cout << setw( 3 ) << *( p + i );
   cout << endl;
}
```

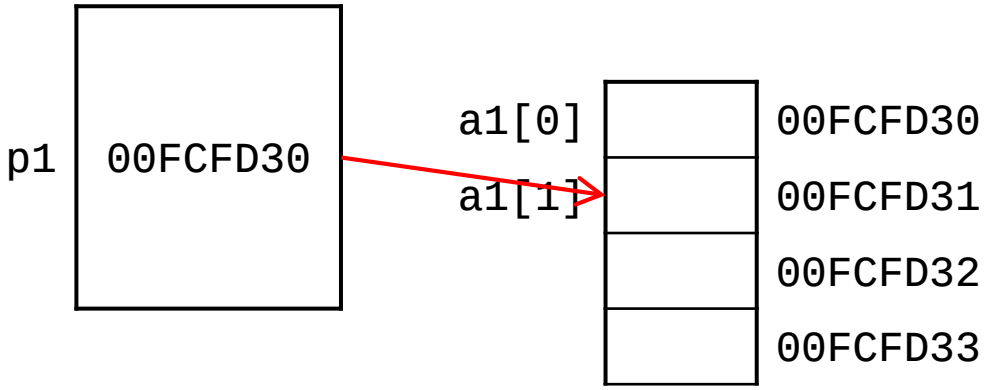
```
  0  2  4  6
  0  2  4  6
  0  2  4  6
  0  2  4  6
```

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```



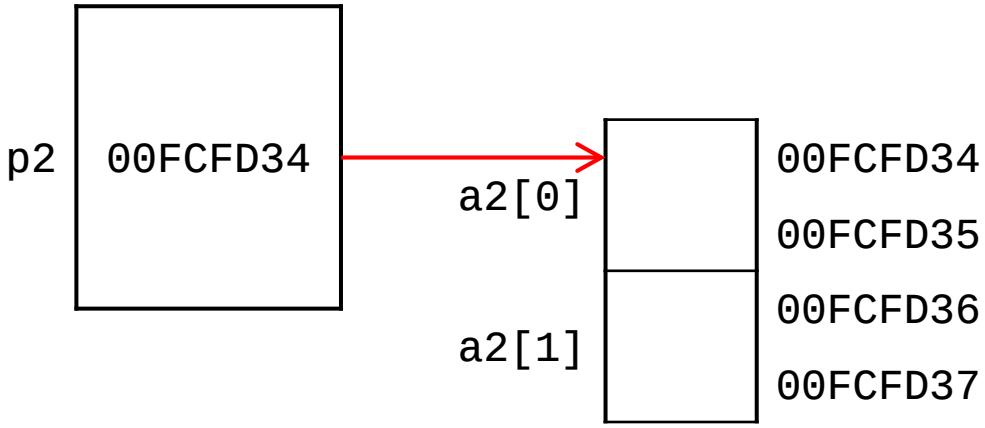
Types of Pointers

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```



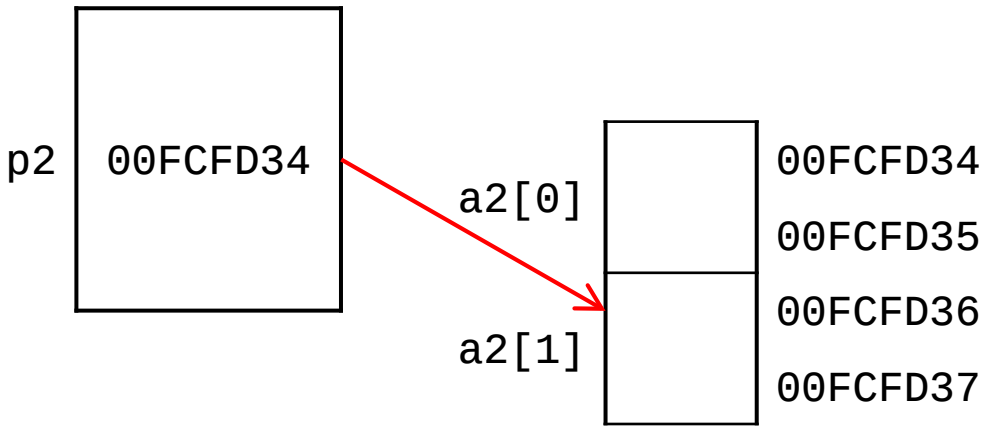
Types of Pointers

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```



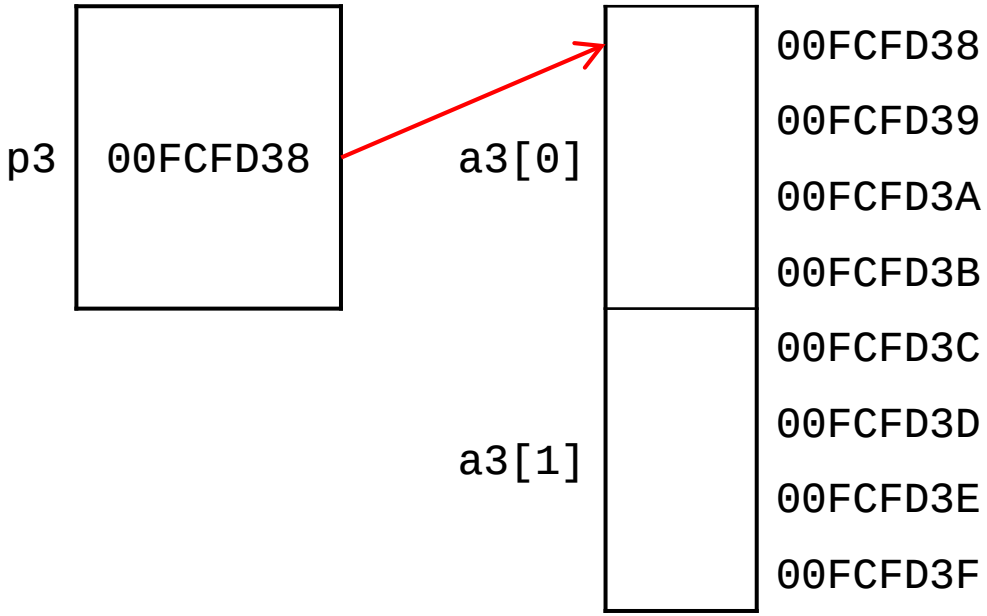
Types of Pointers

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```



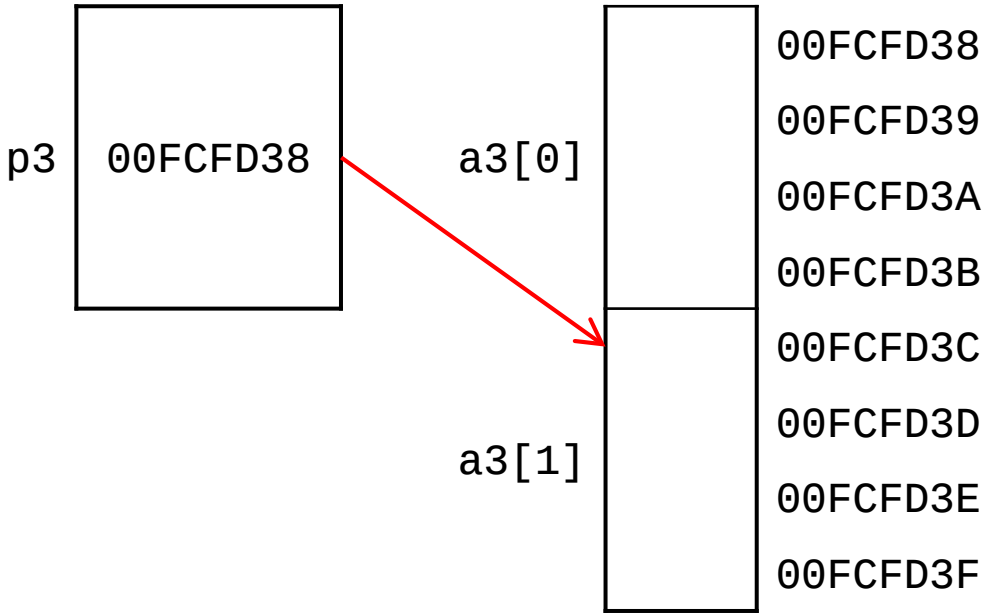
Types of Pointers

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```



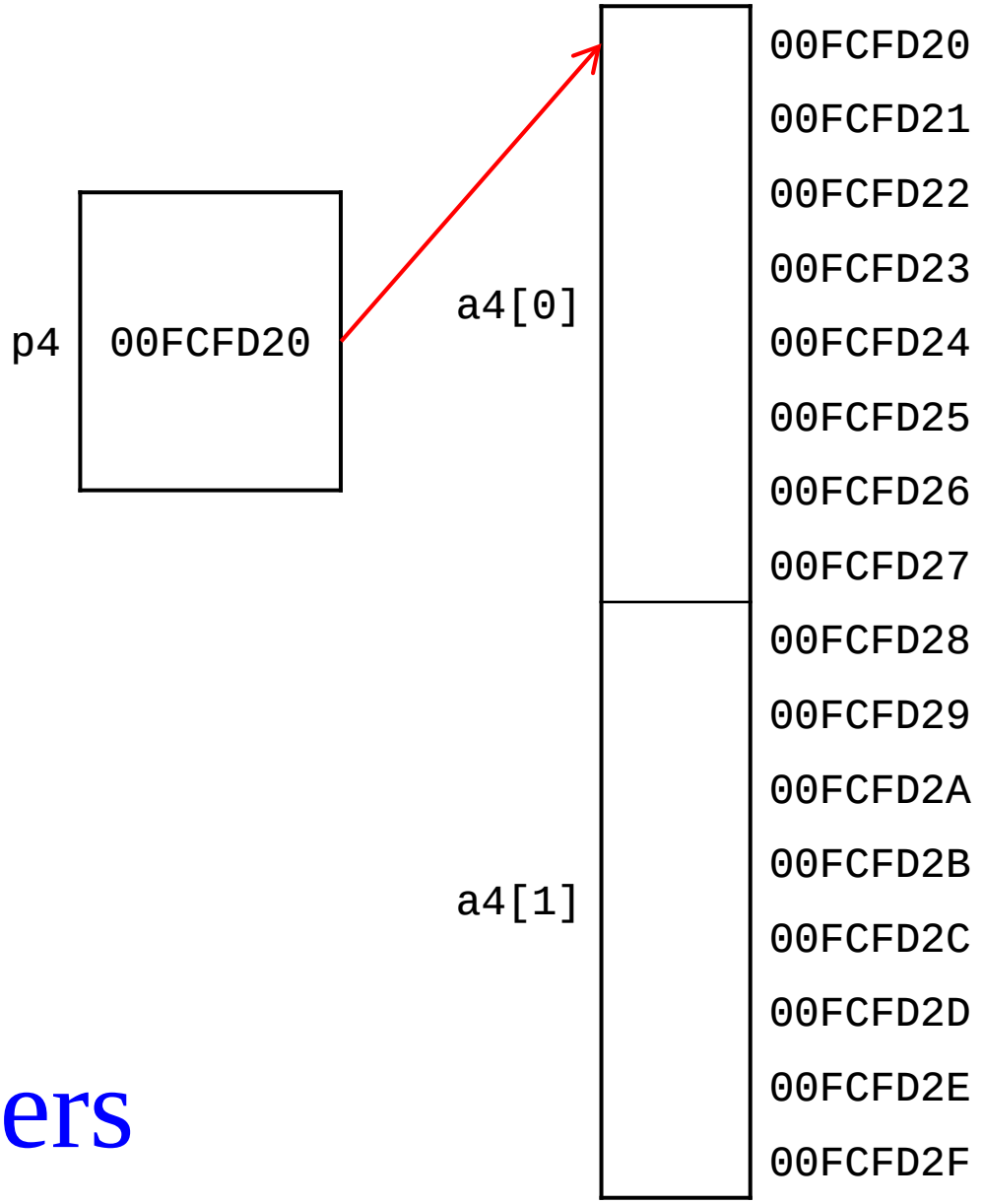
Types of Pointers

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```



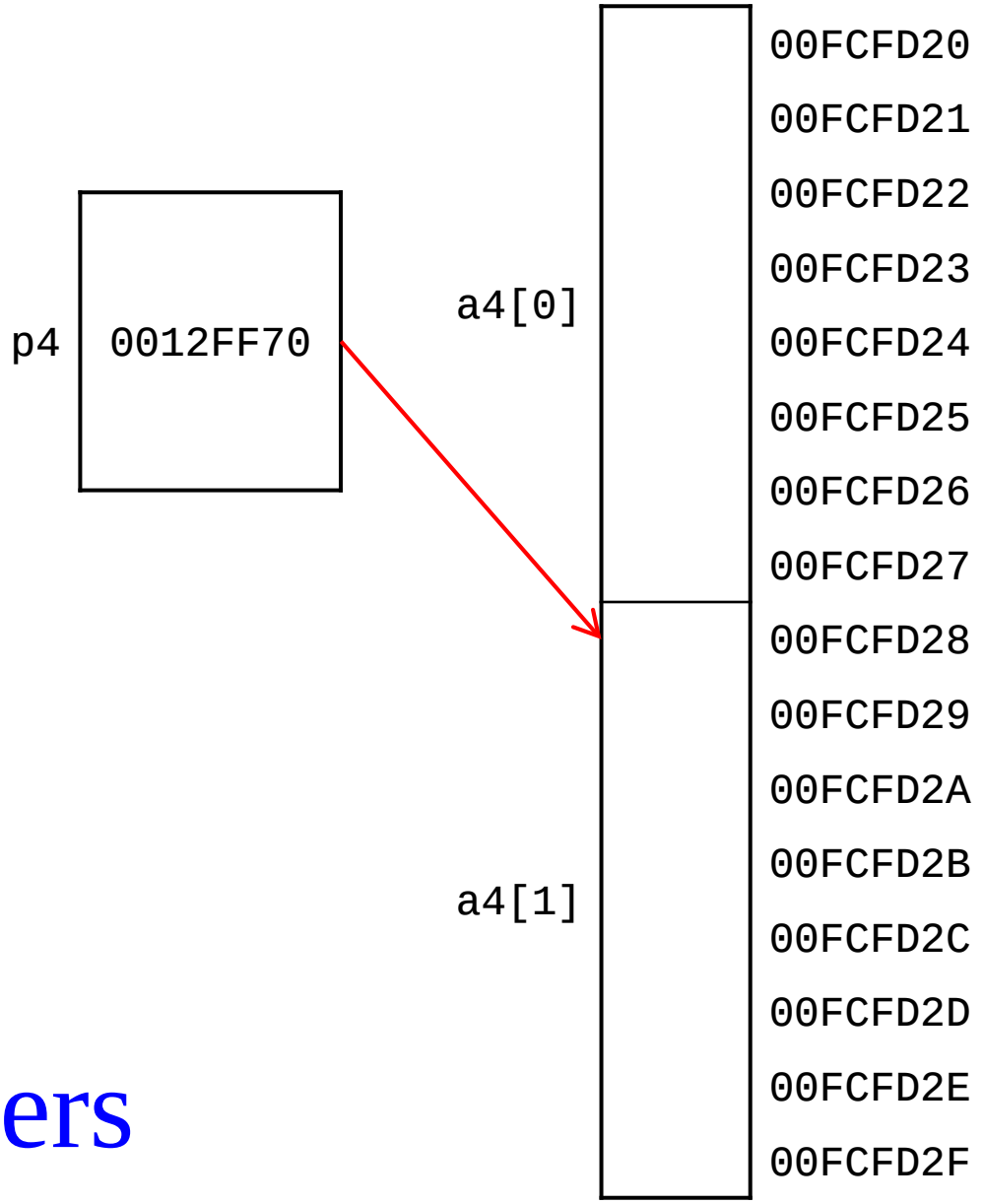
Types of Pointers

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```

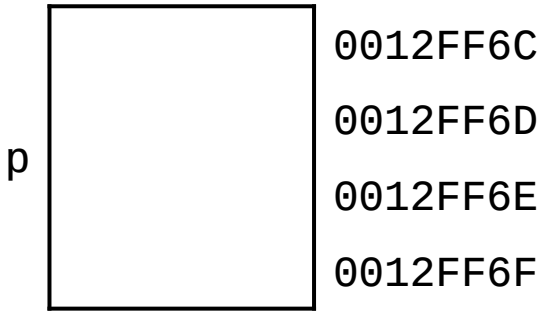


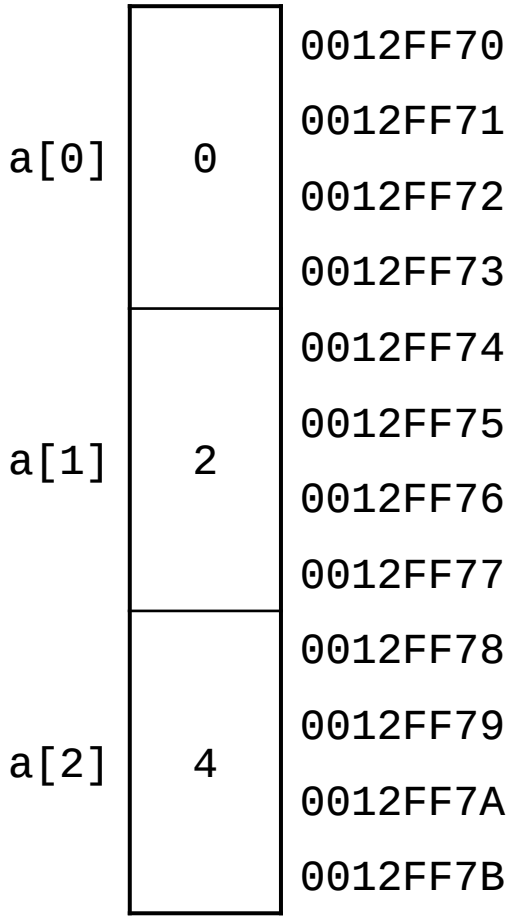
Types of Pointers

```
char a1[ 2 ] = {};
short a2[ 2 ] = {};
int a3[ 2 ] = {};
double a4[ 2 ] = {};

char *p1 = a1;
p1++;

short *p2 = a2;
p2++;

int *p3 = a3;
p3++;

double *p4 = a4;
p4++;
```



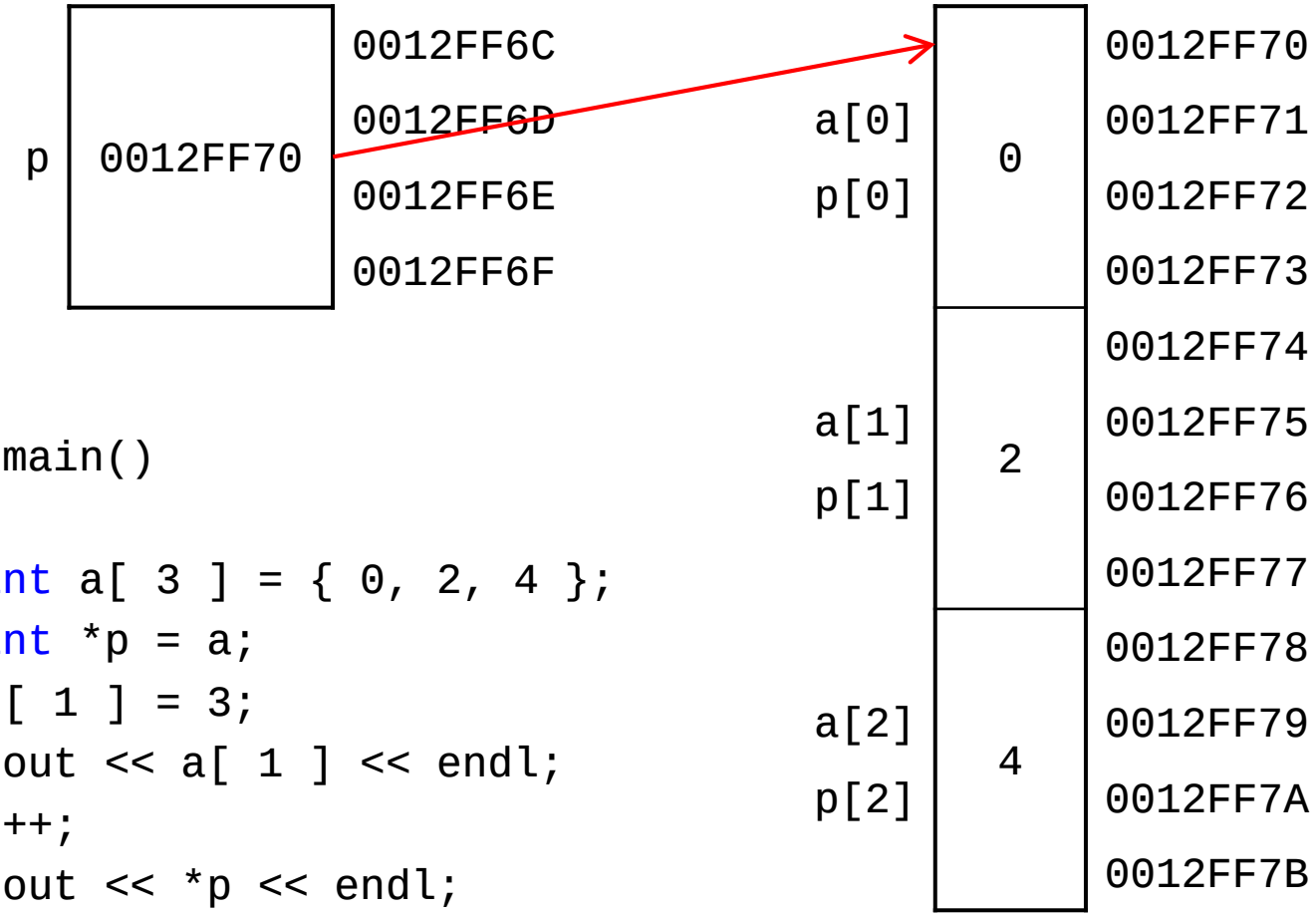
Types of Pointers





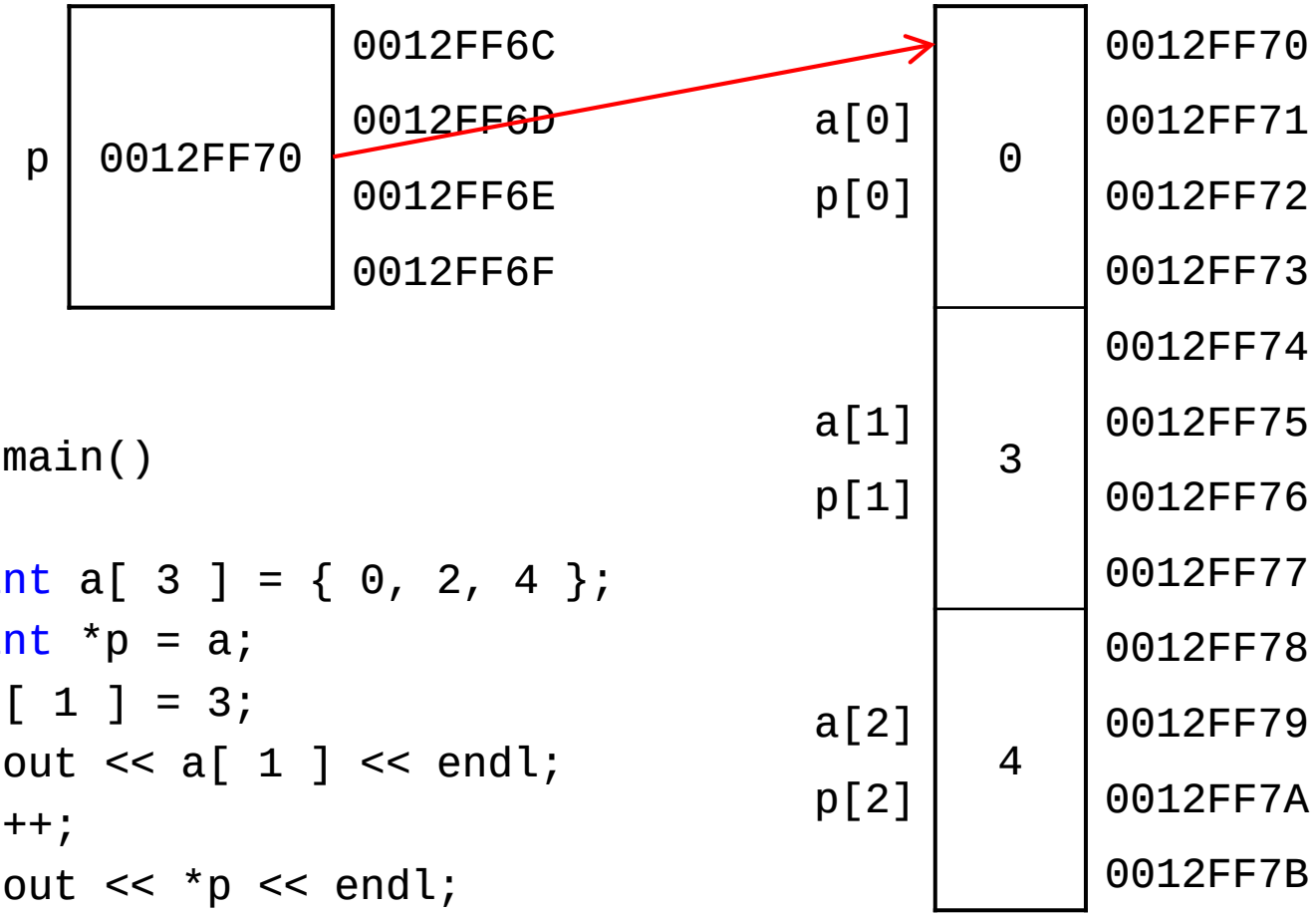

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   int *p = a;
   p[ 1 ] = 3;
   cout << a[ 1 ] << endl;
   p++;
   cout << *p << endl;
}
```

Relationship Between Pointers and Arrays



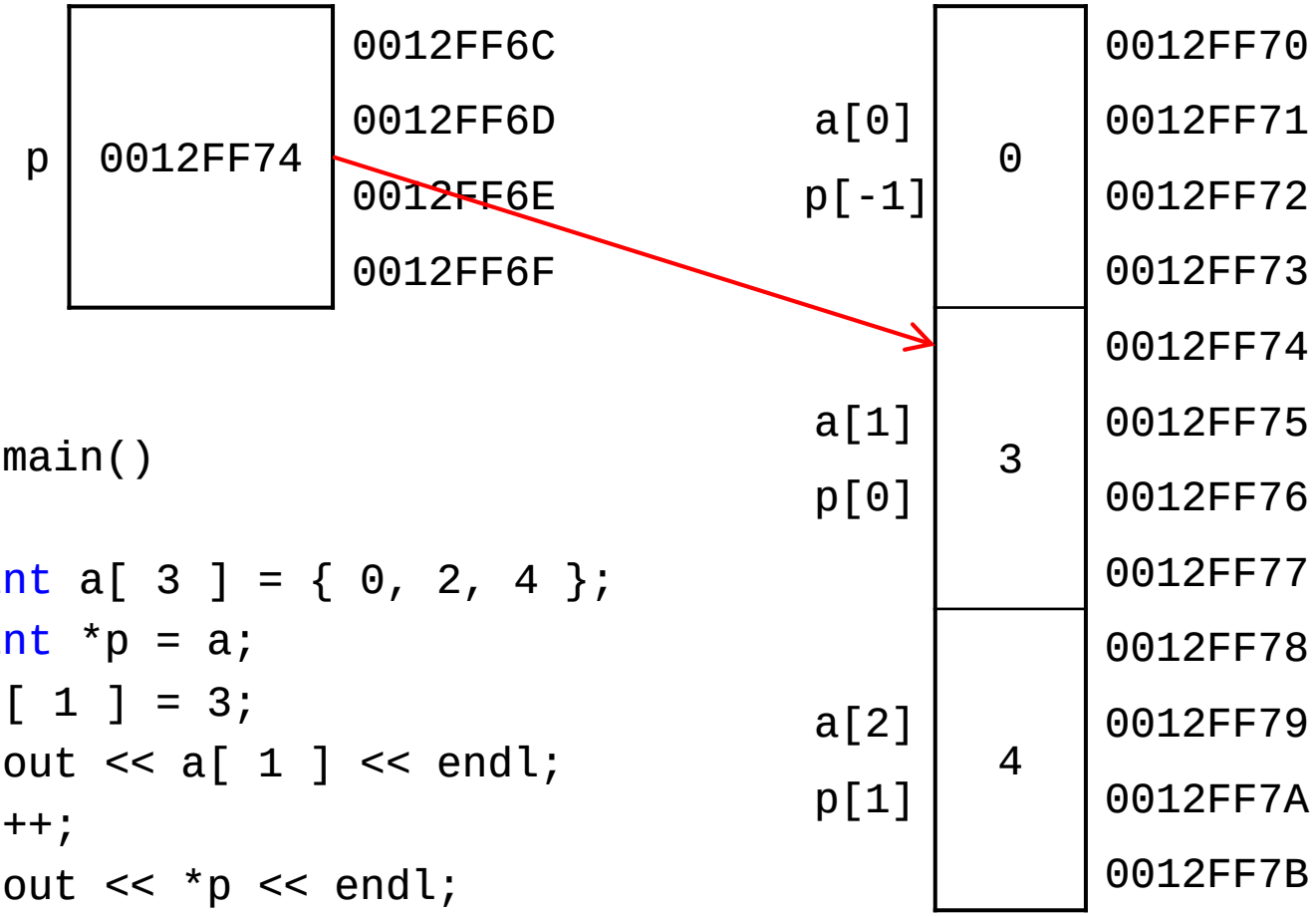

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   int *p = a;
   p[ 1 ] = 3;
   cout << a[ 1 ] << endl;
   p++;
   cout << *p << endl;
}
```

Relationship Between Pointers and Arrays



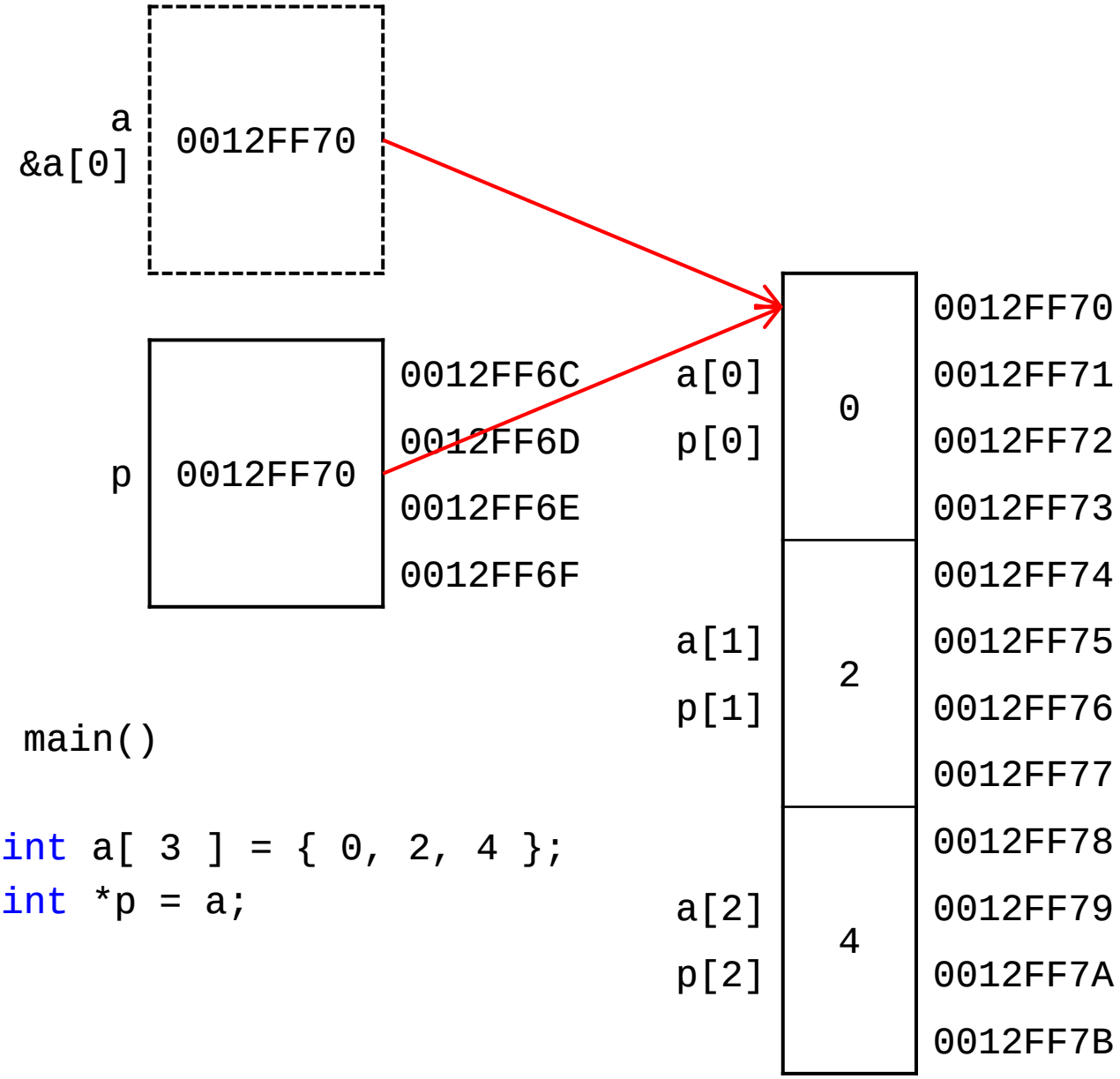

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   int *p = a;
   p[ 1 ] = 3;
   cout << a[ 1 ] << endl;
   p++;
   cout << *p << endl;
}
```

Relationship Between Pointers and Arrays



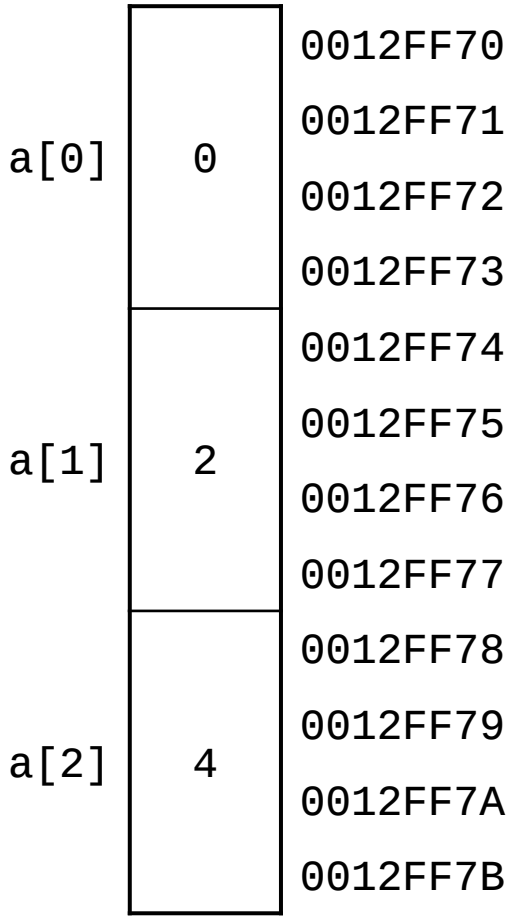

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   int *p = a;
   p[ 1 ] = 3;
   cout << a[ 1 ] << endl;
   p++;
   cout << *p << endl;
}
```

Relationship Between Pointers and Arrays

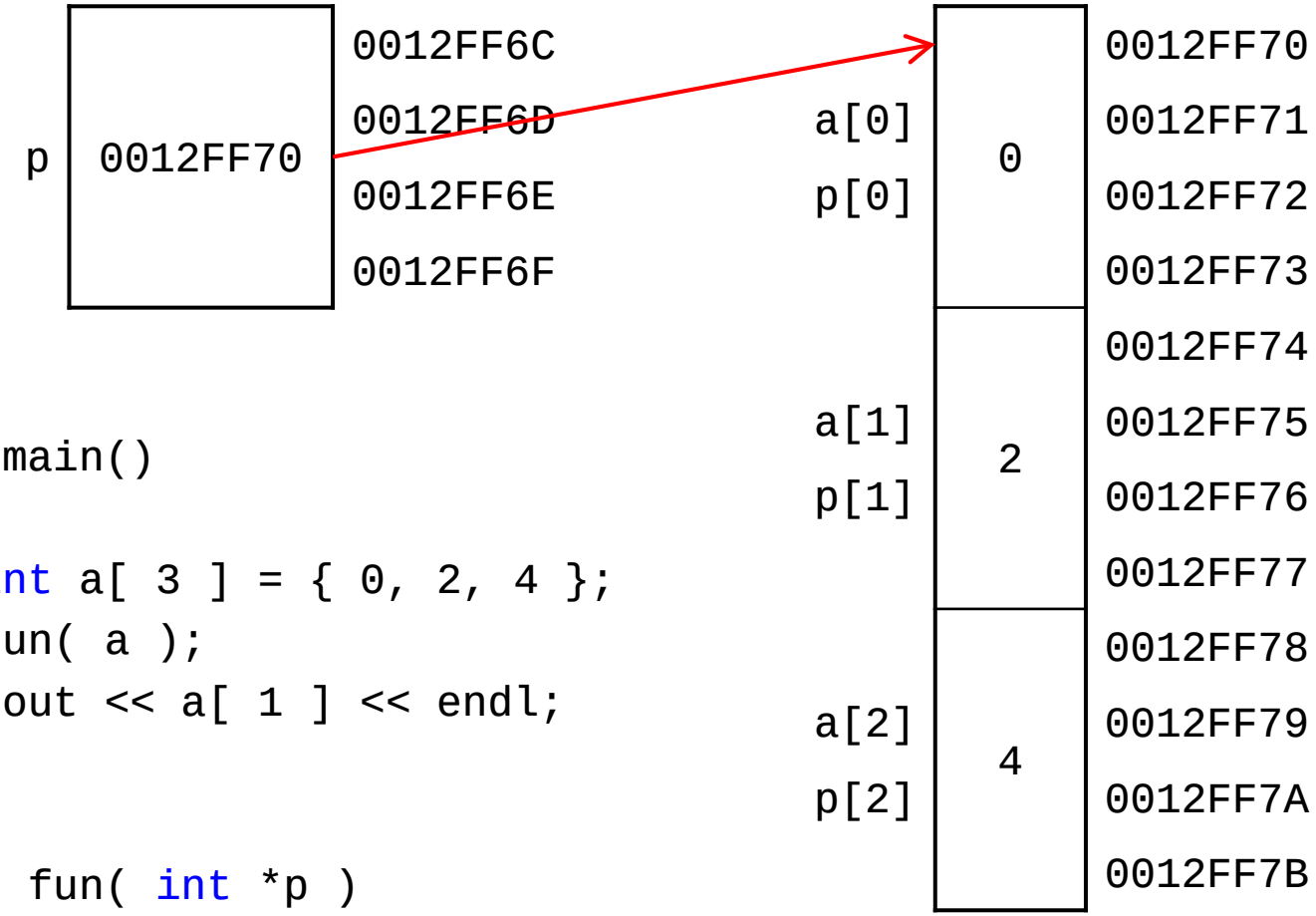



```
int main()
{
    int a[ 3 ] = { 0, 2, 4 };
    int *p = a;
}
```

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int *p )
{
   p[ 1 ] = 3;
}
```

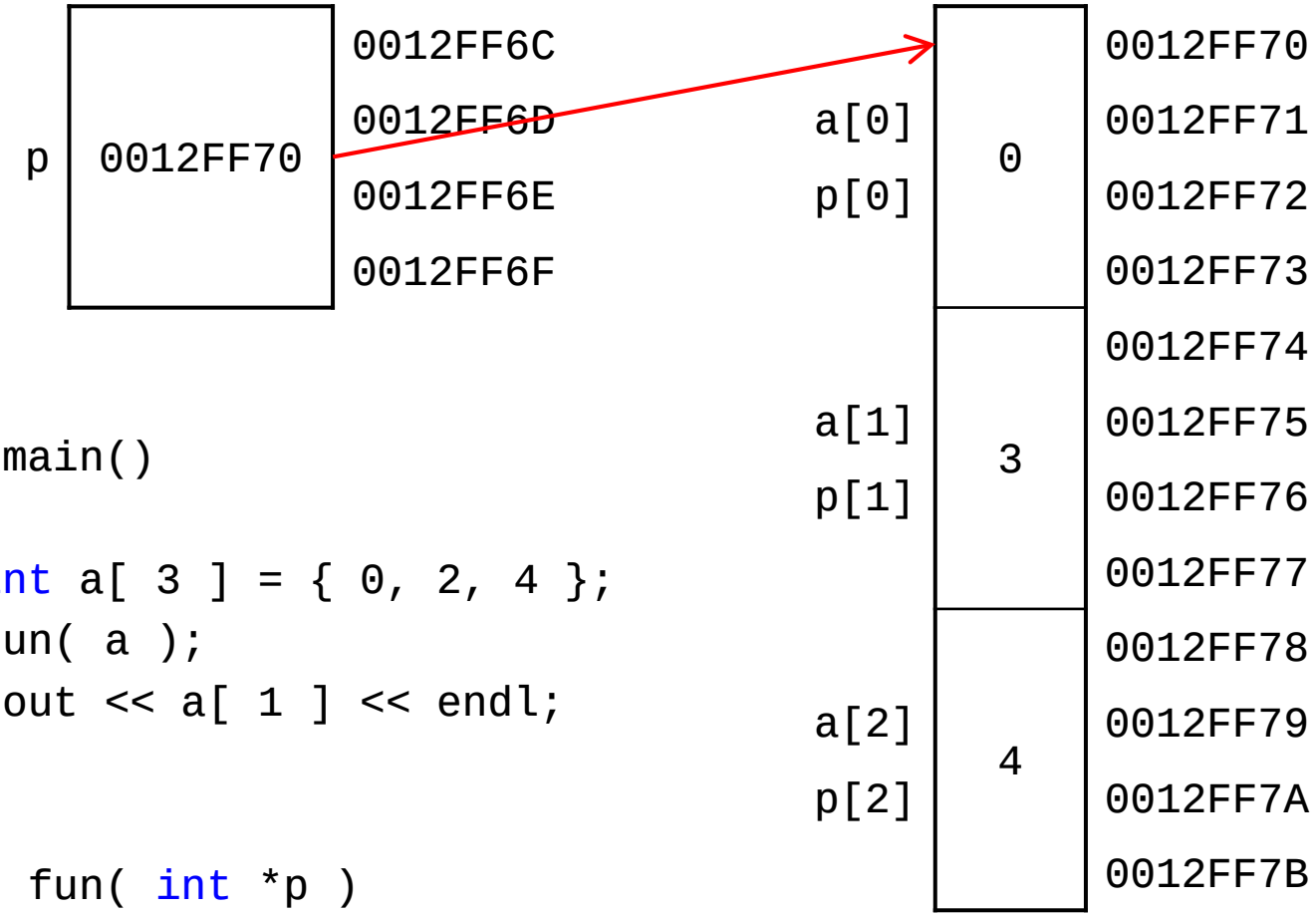


Array Argument




```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int *p )
{
   p[ 1 ] = 3;
}
```

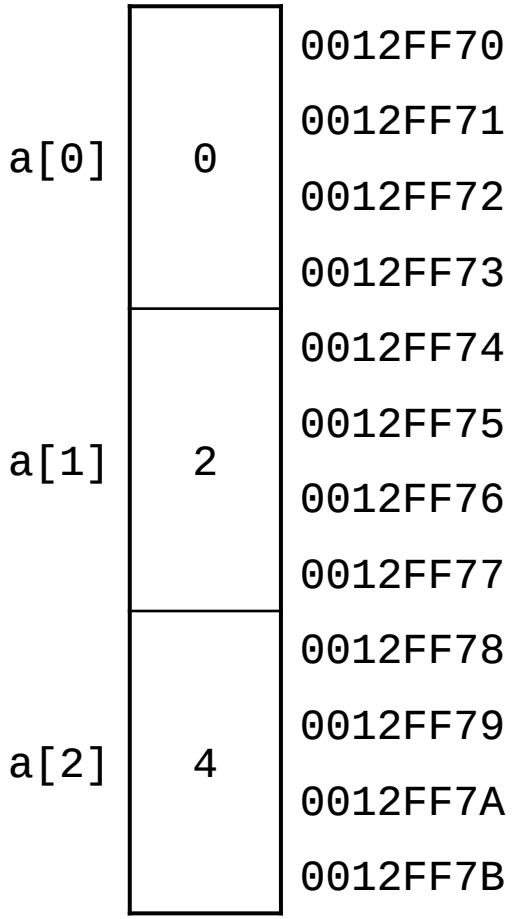
Array Argument



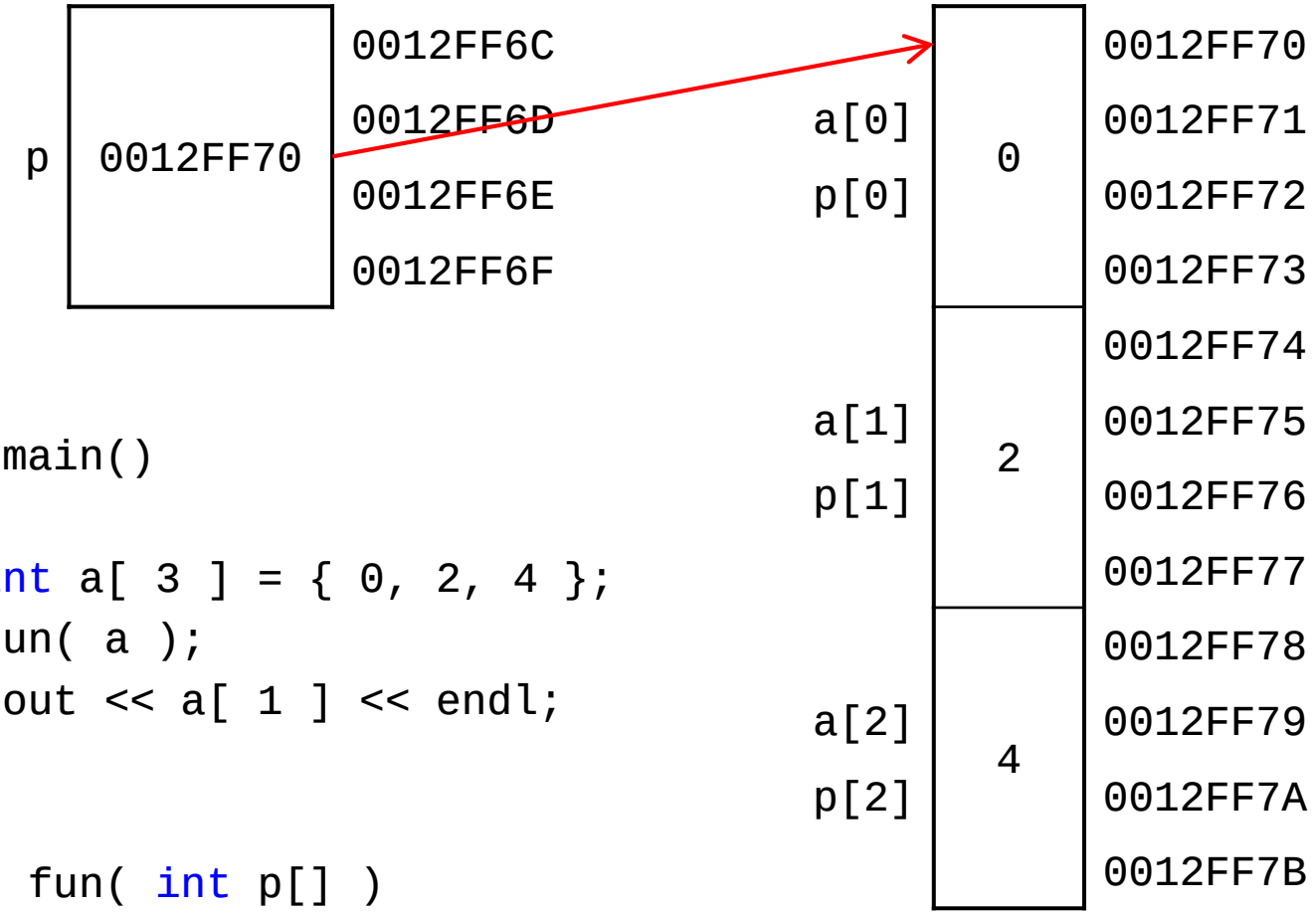

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int *p )
{
   p[ 1 ] = 3;
}
```

Array Argument

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int p[] )
{
   p[ 1 ] = 3;
}
```

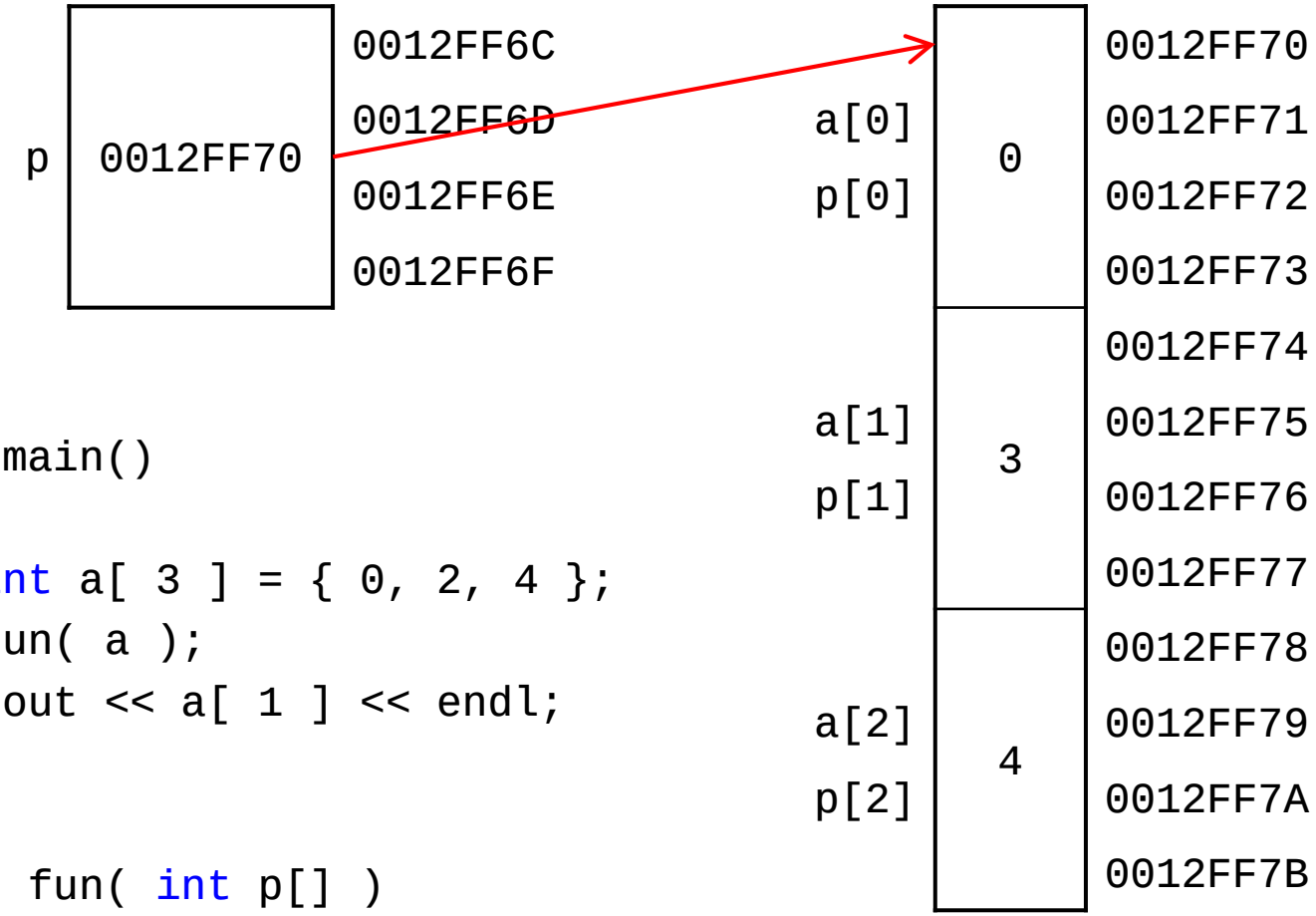


Array Argument




```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int p[] )
{
   p[ 1 ] = 3;
}
```

Array Argument




```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int p[] )
{
   p[ 1 ] = 3;
}
```

Array Argument

Comparison

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int *p )
{
   p[ 1 ] = 3;
}
```

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int p[] )
{
   p[ 1 ] = 3;
}
```

Comparison

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   fun( a );
   cout << a[ 1 ] << endl;
}

void fun( int *p )
{
   p[ 1 ] = 3;
}
```

```
int main()
{
   int a[ 3 ] = { 0, 2, 4 };
   int *p = a;
   p[ 1 ] = 3;
   cout << a[ 1 ] << endl;
}
```

7.7 sizeof Operator

- `sizeof`
  - Unary operator returns size of operand in bytes
  - For arrays, `sizeof` returns

    (size of 1 element) * (number of elements)
  - If `sizeof( int ) = 4`, then

```
int myArray[10];
cout << sizeof(myArray);
```

    will print 40
- `sizeof` can be used with
  - Variable names
  - Type names
  - Constant values

```
1   // Fig. 7.15: fig07_15.cpp
2   // Demonstrating the sizeof operator.
3   #include <iostream>
4   using namespace std;
5
6   int main()
7   {
8      char c; // variable of type char
9      short s; // variable of type short
10     int i; // variable of type int
11     long l; // variable of type long
12     float f; // variable of type float
13     double d; // variable of type double
14     long double ld; // variable of type long double
15     int array[ 20 ]; // array of int
16     int *ptr = array; // variable of type int *
17
```

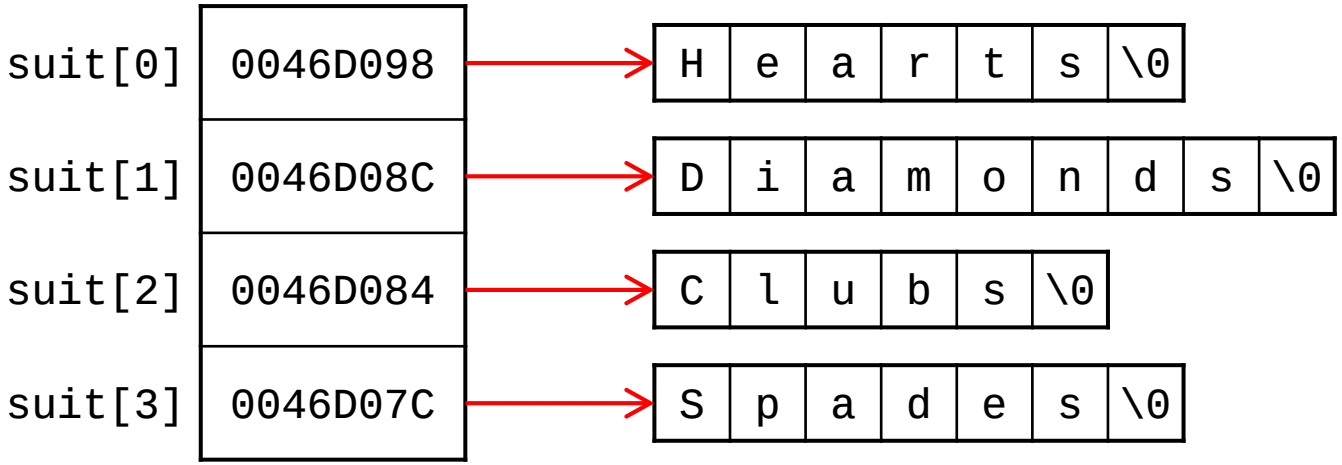
```
18      cout << "sizeof c = " << sizeof c
19           << "\tsizeof(char) = " << sizeof( char )
20           << "\nsizeof s = " << sizeof s
21           << "\tsizeof(short) = " << sizeof( short )
22           << "\nsizeof i = " << sizeof i
23           << "\tsizeof(int) = " << sizeof( int )
24           << "\nsizeof l = " << sizeof l
25           << "\tsizeof(long) = " << sizeof( long )
26           << "\nsizeof f = " << sizeof f
27           << "\tsizeof(float) = " << sizeof( float )
28           << "\nsizeof d = " << sizeof d
29           << "\tsizeof(double) = " << sizeof( double )
30           << "\nsizeof ld = " << sizeof ld
31           << "\tsizeof(long double) = " << sizeof( long double )
32           << "\nsizeof array = " << sizeof array
33           << "\nsizeof ptr = " << sizeof ptr << endl;
34  } // end main
```

```
sizeof c = 1    sizeof(char) = 1
sizeof s = 2    sizeof(short) = 2
sizeof i = 4    sizeof(int) = 4
sizeof l = 4    sizeof(long) = 4
sizeof f = 4    sizeof(float) = 4
sizeof d = 8    sizeof(double) = 8
sizeof ld = 8   sizeof(long double) = 8
sizeof array = 80
sizeof ptr = 4
```

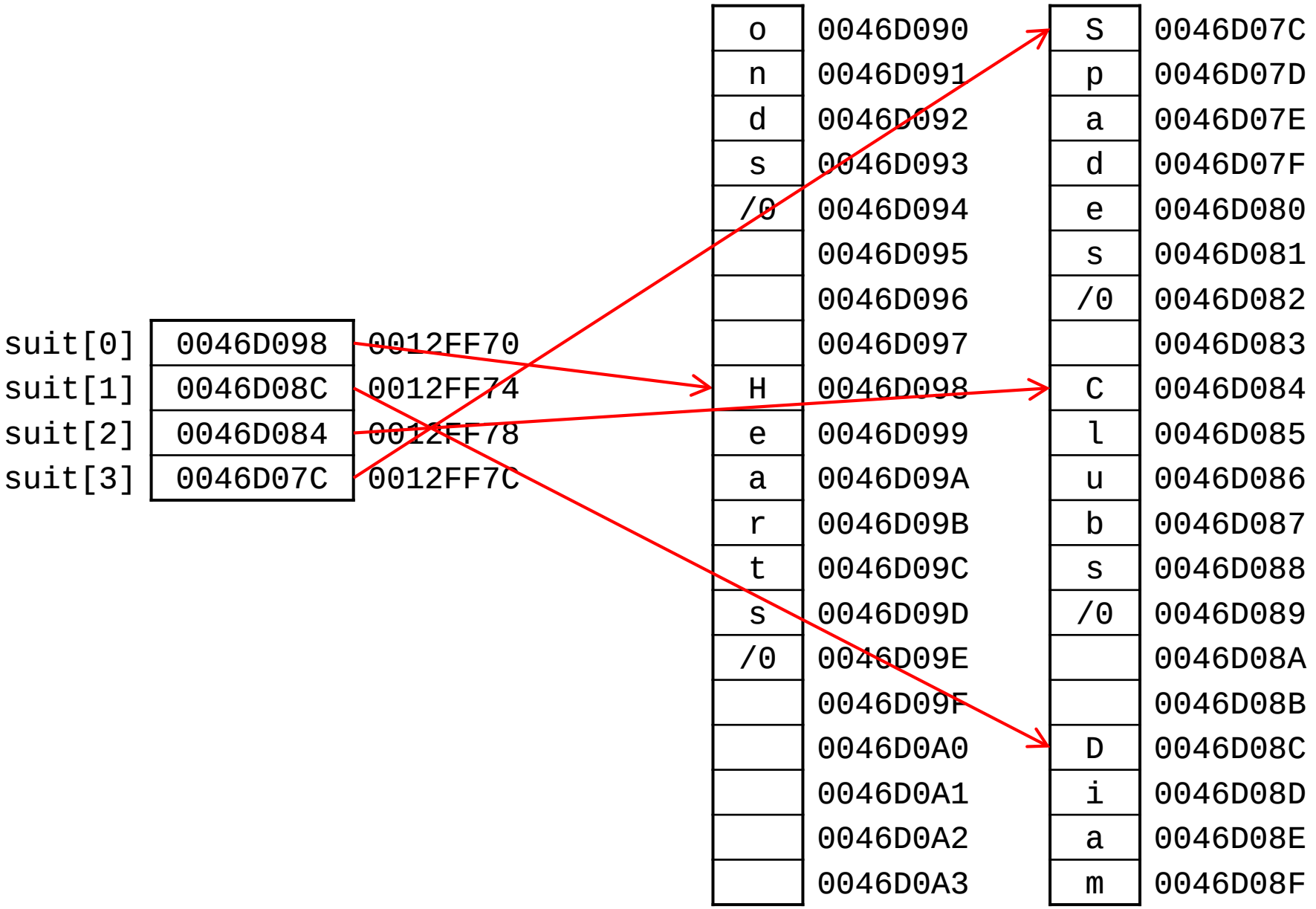
Arrays of Pointers

- Arrays may contain pointers.
- A common use of such a data structure is to form an array of pointer-based strings, referred to simply as a string array.
- Each entry in an array of strings is simply a pointer to the first character of a string.

```
const char * const suit[ 4 ] =
   { "Hearts", "Diamonds", "Clubs", "Spades" };
```



```
char *suit[4] = { "Hearts", "Diamonds", "Clubs", "Spades" };
```



```
1  // Fig. 7.10: fig07_10.cpp
2  // Attempting to modify data through a
3  // nonconstant pointer to constant data.
4
5  void f( const int * ); // prototype
6
7  int main()
8  {
9     int y;
10
11    f( &y ); // f attempts illegal modification
12 } // end main
13
14 // xPtr cannot modify the value of constant variable to which it points
15 void f( const int *xPtr )
16 {
17    *xPtr = 100; // error: cannot modify a const object
18 } // end function f
```

```
1  // Fig. 7.11: fig07_11.cpp
2  // Attempting to modify a constant pointer to non-constant data.
3
4  int main()
5  {
6     int x, y;
7
8     // ptr is a constant pointer to an integer that can
9     // be modified through ptr, but ptr always points to the
10    // same memory location.
11    int * const ptr = &x; // const pointer must be initialized
12
13    *ptr = 7; // allowed: *ptr is not const
14    ptr = &y; // error: ptr is const; cannot assign to it a new address
15 } // end main
```

```
1  // Fig. 7.12: fig07_12.cpp
2  // Attempting to modify a constant pointer to constant data.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     int x = 5, y;
9
10    // ptr is a constant pointer to a constant integer.
11    // ptr always points to the same location; the integer
12    // at that location cannot be modified.
13    const int *const ptr = &x;
14
15    cout << *ptr << endl;
16
17    *ptr = 7; // error: *ptr is const; cannot assign new value
18    ptr = &y; // error: ptr is const; cannot assign new address
19 } // end main
```